POLYNOMIAL PERCEPTRON AND IT's USE IN THE RECOVERY OF NOISY BIOELECTRIC SIGNALS

by

JAI PRAKASH SAINI

EE
1996
M
SAI
POL

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JANUARY, 1996

POLYNOMIAL PERCEPTRON AND IT's USE IN THE RECOVERY OF NOISY BIOELECTRIC SIGNALS

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Jai Prakash Saini

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

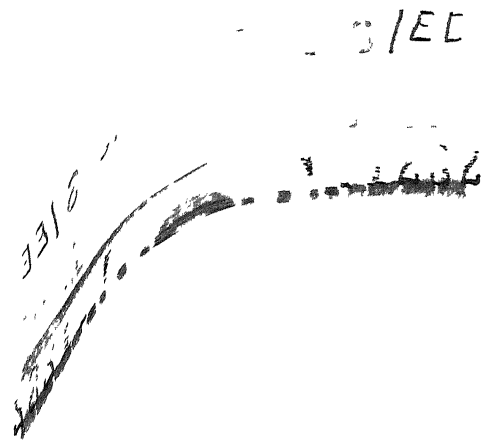
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

January 1996

Certificate

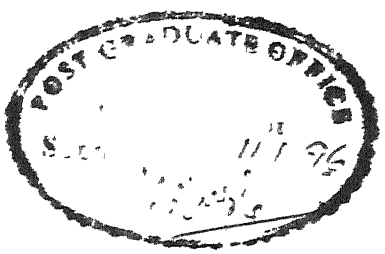
It is certified that the work contained in the thesis entitled **POLYNOMIAL PERCEPTRON AND IT's USE IN THE RECOVERY OF NOISY BIOELECTRIC SIGNALS** by Jai Prakash Saini, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree

Dr G C Ray
Professor
Department of Electrical Engineering
I I T Kanpur



A121282

Certificate



It is certified that the work contained in the thesis entitled **POLYNOMIAL PERCEPTRON AND IT's USE IN THE RECOVERY OF NOISY BIOELECTRIC SIGNALS** by Jai Prakash Saini, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree

Dr G C Ray
Professor
Department of Electrical Engineering
I I T Kanpur

Acknowledgement

I would take this opportunity to express my sincere gratitude towards Dr G. C Ray my thesis supervisor for having lit in my heart the flame symbolising the spirit of enquiry for having encouraged me to make an attempt against all odds and for standing by me in my moments of sorrow and joy

I thank my mother, my wife, my brothers and my son for their staunch support

I would cherish my association which I had forged here with Sanjay G Joshi. Samarendra Dandapat, M R Warsi, Bhaskar Bhar Suvrat Gupta, Vikram Singh. Sanjay Gupta. Prakash Veer, Hemant, Mukul Katiyar and everybody else in the communication and signal processing stream who were a great support during my thesis and stay at I I T Kanpur

Finally, I would be failing in my duty if I do not express my gratitude towards Dr M. C Srivastava, Head, Electronics Engg Dept., K N.I.T. Sultanpur and Director K N.I.T. Sultanpur who have afforded me this great opportunity

Jai Prakash Saini

Dedicated To

My Parents

Abstract

We have investigated the problem of reconstructing the ECG signal, which is corrupted with additive noise The noise appears when the signal passes through a dispersive channel such as in long distance medical telemetry system We have used neural network for suppression of noise Polynomial perceptron and fractionally spaced recursive polynomial perceptron are used as nonlinear classifiers to reconstruct the ECG signals The behaviour of backpropagation algorithm and complex backpropagation algorithm is investigated It is shown that the both algorithms are powerful, but faces some practical difficulties The selection of learning parameters number of hidden layers and number of nodes in each hidden layer are experimental Polynomial perceptron is the alternative nonlinear architecture to approximate the optimal equalizer solution The behaviour and nonlinear mapping ability of polynomial perceptron and fractionally spaced bilinear perceptron are investigated It is shown that these techniques can approximate any continuous function within a specified accuracy The manner in which these neural network algorithms can be utilized is described. The complex neuron structure is used to modify the above methods for complex input sequences The performances of these two methods are compared and their relative features and limitations are discussed The applications of these methods to 16-level quadrature amplitude modulation are considered for reconstruction of ECG signals Simulation results suggest that the fractionally spaced bilinear perceptron network recover the ECG signal in the high noise environment

Contents

Chapter 1

Introduction

There are many expert systems which have been developed in the last few years in an attempt to solve medical diagnostic problems automatically We have investigated that the artificial neural networks may be a better solution for reconstructing the ECG (Electrocardiogram) signals which are corrupted with noise and other interference in long distance telemetry

In recent years neural networks have been proposed as solutions to a variety of problems in the areas such as signal processing including image processing pattern recognition identification, prediction and control of dynamical systems The pattern recognition by the neural networks are useful in many practical problems such as medical diagnosis, speech recognition and adaptive control The purpose of any attempt at pattern recognition is the identification of the underlying characteristics which are common to a class of objects The correct identification of the underlying characteristics enables one to extrapolate, i e, to identify a new object as belonging to a certain class on the basis of its underlying characteristics and in spite of the variations of incidential characteristics within the class

1.1 Medical telemetry

Medical telemetry systems transmit signals continuously or at recurrent intervals and measure the variables which are displayed recorded, or used to activate a control mech-

anism.

Medical telemetry may be a combination of wired and wireless For example a wireless system may send data from the patient to the hospital console Then at the console data may be triped over the telephone lines In wireless telemetry, the carrier is a signal which carries the data through the air The data themselves modulate this carrier Modulation refers to a nonlinear operation in which message data are used to modify some characteristic of the carrier The carrier is usually sinusoidal or periodic sequence of pulses Some times multiplexing is also preferred with telemetry systems to transmit the data from different patient transducers simultaneously

The most urgent application of medical telemetry is the immediate monitoring of an unstable coronary condition upon the onset of distress symptoms A specialist in the city or on the other side of the earth can make the immediate assessment of the onset of myocardial infarction Most myocardial-infarction patients experience either bradycardia tachycardia heart block or excessive premature ventricular contractions before reaching the hospital These arrythmias precede cardiac arrest. killing 60 percent of heart attack victims before reach the hospital

When a patient, having a pacemaker installed has cardiovascular surgery or treatment or has a chronic problem. there is often a need for frequent outpatient ECG data retrieval Normally this application requires that ECG data be transmitted from the patient's home to the doctors's office or to the hospital This is an ideal application of medical telemetry

All wired and wireless systems need both transmitter and receiver In multiplexing. several signals are placed on one carrier Signal conditioners convert and amplify the data and match the incoming signal to the multiplexer or transmitter A block diagram of typical single channel system is shown in fig 1 1 The major difficulty which arises in biotelemetry is the interference with other systems. Therefore, after demodulation. channel equalizer is required to suppress the noise and interference which gets added during transmission of data. The neural network may used as the channel equalizer to suppress the noise and interference

We have focussed our attention on the part of channel equalization using neural networks

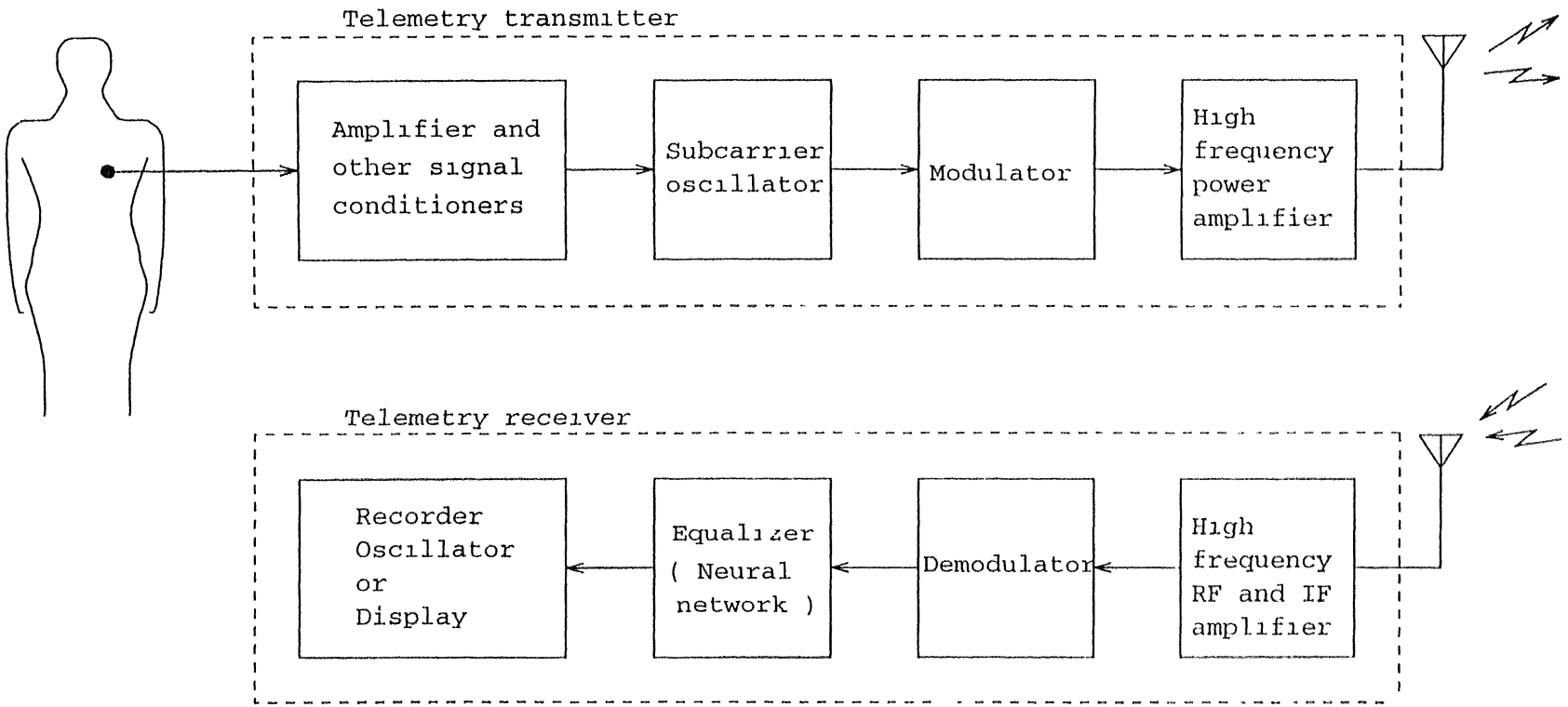


Fig 1 1 Typical single channel telemetry system

1.2 Artificial neural network

The most important feature of artificial neural network is that they perform a large number of numerical operations in parallel These operations involve simple arithmatic operations as well as nonlinear mapping and computation of derivatives The computational elements used in the neural networks are nonlinear and typically analog. these elements are connected via weights that are typically adapted during use to improve the performance Almost all data stored in the network are involved in recall computation at any given time The distributed neural processing is typically performed within the entire array composed of neurons and weights This indicates that parallel distributed computing makes efficient use of stored data (weights) and of the input data. Therefore, we can resemble the artificial neural network similar to human brain performance in two respects

- The knowledge is acquired by the network through the learning process.
- The interneuron connection strengths known as synaptic weights are used to store the knowledge

The knowledge refers to stored information or models used to interpret predict and approximately respond to the output. The adaptation is a major focus on neural network. which provides a degree of robustness by compensating for minor variabilities in characteristics of processing elements Therefore, the modification of synaptic weights provides the traditional method for the design of the neural networks. The majority of the works on the neural networks is directed towards a better architecture for pattern classification

1.3 Review

The number of networks whose architectures are found to be useful for pattern classification have been proposed by the several authors Here, we discuss briefly about the various architectures

In 1943, for the first time, McCulloch and Pitt proposed the standard architecture [17]

[18] of neuron as follows

$$y = f\left(\sum_{i=0}^{n} w_i x_i\right), \qquad i = 1, 2, \quad , n \tag{1 1}$$

where x_i are the the inputs to neuron x_0 is taken as the threshold input w_i are the synaptic weights and y is the output of the neuron The function $f(\ )$ is nonlinear which can be either hard limiter, threshold logic, sigmoid or hyperbolic tangent type function There is no a priori reason to assume that the synaptic weights are constants This is the first approximation to the complex behavior of a neuron introduced by McCulloch and Pitt

Williams-Zipser [6] proposed the architecture which was not purely feedforward This architecture permits any neuron in the network may be connected to any other neuron in the network

Jordan-Elman [5] proposed the architecture which has the extra layer of neurons that copy the current activations in the hidden layer neurons and after delaying these values for one time instant, feedback them as additional inputs into the hidden neurons

Lapedes-Farber [5] proposed an architecture which is defined as follows

$$y = f(a(t)) \tag{1 2}$$

where the activation function a(t) is the linear combination of the input to the neuron Output is a nonlinear function of the activation function The architecture is shown in fig 1 2 If the input derived from the previous layer only, then there is no feedback around the neuron and the architecture is feedforward only Alternatively, there are various possibilities for placing the feedback paths within network When the feedback is incorporated as local activation feedback as shown in fig 1 3, the input $x_i = a(t - i). i =$ 1 2, n is delayed version of the activation or the combination of delayed version of the activation and the delayed version of the input variables When each synapse incorporate a feedback structure, the local synapse feedback architecture results, as shown in fig 1 4

This feedback architecture may be in the form of the IIR or FIR filter and each synapse function may be modified by a linear transfer function When the feedback is around the nonlinear function as shown in fig 1 5, then the input will be the delayed version of

output values i e, $x_i = y(t-i)$ This type of feedback architecture is known as the local feedback architecture This architecture has the feedback after the nonlinear function while the local activation feedback and local synapse feedback have the feedback before the the nonlinear function, therefore the behaviour of this architecture is different from other two architecture

Back-Tsoi [5] proposed the locally recursive global feedforward network with local synapse feedback which are called as IIR synapse multilayer perceptron Back-Tsoi architecture is similar to local synapse architecture as shown in fig 1 4, where the synaptic weights are replaced by the linear transfer function with poles and zeroes This is a modified version of the McCulloch-Pitt neuron The architecture is follows

$$y(t) = f\left(\sum_{i=0}^{n} G_i z^{-1}\right) x_i(t), \qquad i = 1, 2, \qquad n \tag{1 3}$$

where,

$$G\left(z^{-1}\right) = \frac{\sum_{j=0}^{m_z} a_j z^{-j}}{\sum_{j=0}^{n_p} b_j z^{-j}} \tag{1 4}$$

is a linear transfer and $z^{-i}x(t)$ is defined as $x(t-i)$, If feedback is taken from the previous layer, then it is called as local synapse feedback and if it is derived from the output $y(t)$, it is called as global output feedback

Frasconi-Gori-Soda [7] [5] proposed two architectures namely local activation feedback architecture and locally recurrent globally feedforward architecture with a feedback path being taken from the output of the hidden layer neuron The generalized architecture of Fracsconi-Gori-Soda is shown in fig 1 6, where, D is the unit time delay This architecture is different from the Back-Tsoi architecture in that Back-Tsoi have taken the local feedback before nonlinearity, while Frasconi-Gori-Soda has taken the local feedback after it has passed through the nonlinearity This architecture is defined as follows

$$y(t) = f\left(\sum_{i=1}^{m} k_i y(t-i) + \sum_{i=0}^{n} w_i x_i(t)\right) \tag{1 5}$$

where, k_i are the constants When the feedback is taken from the output of the same neuron, it is called as local feedback and in the case when feedback is taken from the other neurons, global recurrent networks results.

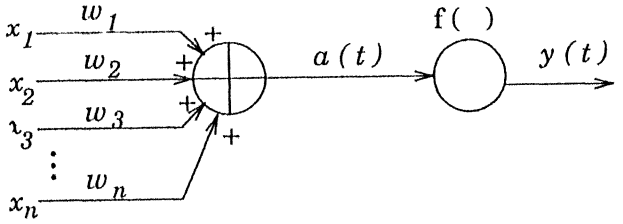


Fig 1.2 Lapedes-Farber architecture

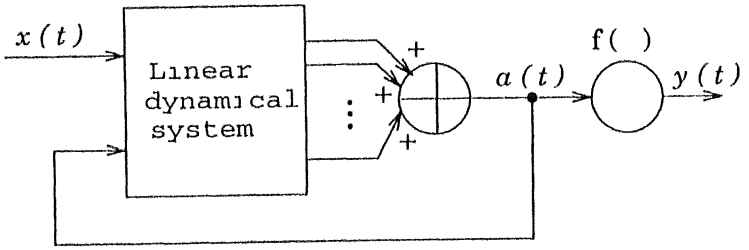


Fig 1.3 Local activation feedback architecture

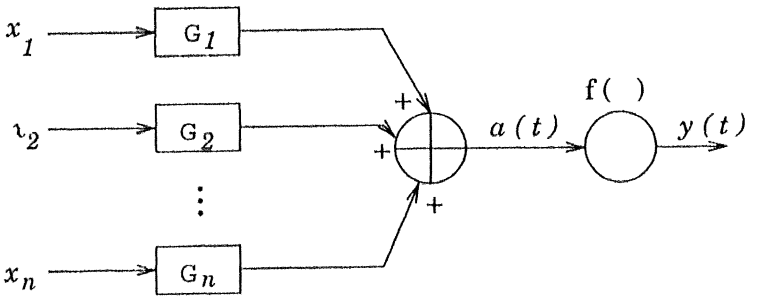


Fig.1 4. Local synapse feedback architecture

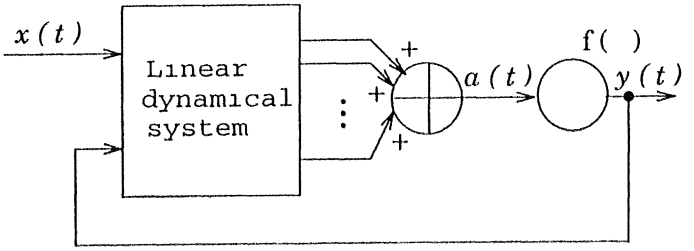


Fig 1 5 Local output feedback architecture

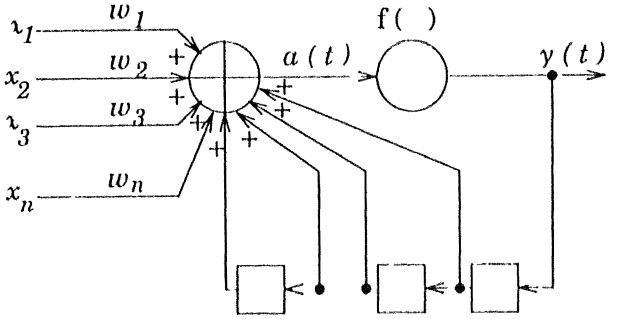


Fig 1 6 Frasconi-Gori-Soda architecture

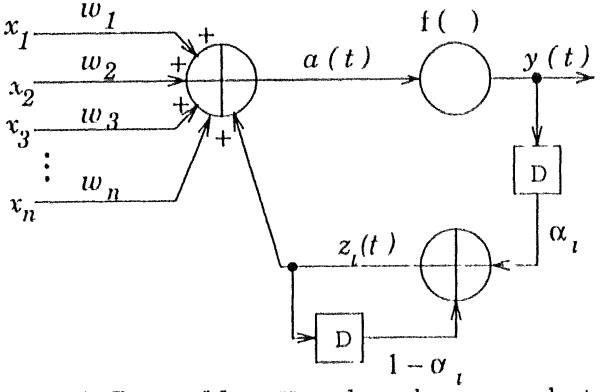


Fig 1 7 Poddar-Unnikrishnan architecture

D Vries and Principe [5] proposed the following architecture

$$\frac{du(t)}{dt} = -au(t) + g_0 y(t) + \int_0^t g(t-s)y(s)ds + x(t) \tag{1.6}$$

where u(t) is the activation state with $y(t) = f(u(t))$, g_0 is the gain constant controlling the dynamics of the system and $g(t)$ is the convolution kernel In this architecture the synapse is modeled by the convolution operator They also proposed a discrete time model, where time delay is modelled by the operator z^{-1} and $z^{-1}x(t)$ is $x(t-1)$

Poddar-Unnikrishnan [5] proposed the architecture with neurons that can have memory, to store information regarding past activations of network neurons The architecture as shown in fig 1 7 is defined as follows

$$y(t) = f\left(\sum_{i=0}^{n} w_i x_i(t) + \sum_{i=0}^{n} k_i z_i(t)\right) \tag{1.7}$$

The output of the memory neuron from previous layer is defined as follows

$$z_i(t) = \alpha y(t-1) + (1-\alpha)z_i(t-1) \tag{1.8}$$

where, α is the constant The memory neuron remembers the past output values to that particular neuron In this case, the memory is taken to be in the form of an exponential filter. The Poddar-Unnikrishnan architecture can be c sidered as a special case of the generalised Frasconi-Gori-Soda architecture

The above architectures may be considered as the generalization of the classical McCulloch-Pitts neurons from input output point of view Instead of constant synaptic weights, the architectures have the synaptic weight that are modelled by the linear transfer function The feedback from the output is also modelled by the linear transfer function Therefore, these generalized neuron can then be used with multilayer perceptron

There are, however, some of major disadvantages associated with this highly nonlinear structure that restrict the multilayer perceptron for practical application The selection of architectures and parameters for multilayer perceptron is mainly by experiment, and its structure is complicated The convergence rate of multilayer perceptron is rather slow, and its learning time is typically very long

There is another alternative nonlinear technique known as polynomial perceptron [8] [9] [4] [10] [11], which is theoretically more tractable compared with the multilayer perceptron In this technique the polynomial approximation method is employed as a means of approximately realising the optimal solution It is capable of forming arbitrary complex nonlinear decision surface to approximate any continuous function within arbitrary specified accuracy, which is similar to the multilayer perceptron

D F Specht [8] first time proposed the idea for pattern classification using polynomial functions Specht referred to his network as the polynomial discriminant method and showed that the polynomial discriminant method is simple for determining the weight coefficients for cross product and power terms in variable input The method was based on the nonparametric estimation of a probability density function for each category to be classified and the bayes decision rule was used for the classification Because polynomials can represent functions that are far more complex than any linear function, the polynomial discriminant function is not restricted to linear seperable problems Specht applied this method successfully to the problem of automatic analysis of ECG signals

D F Specht [9] present the review of two methods namely probablistic neural network and polynomial adaline for classification based on the bayes strategy and nonparametric estimators for probabilty density function The performances of these two methods are compared with multilayer perceptron and the relative advantages and disadvantages are also discussed in this paper

Chen-Gibson-Cown [4] investigated the application of simple nonlinear polynomial perceptron structure and showed that approximate realisation of the optimal equalisation solution can be implemented using a polynomial perceptron

Xiang-Bi-Le-Ngoc [11] proposed the idea of fractionally spaced recursive polynomial perceptron and gave the architecture of bilinear perceptron, which is based on the model of bilinear systems [13] The bilinear systems are linear seperately with respect to the state (input) and the control (output), but nonlinear jointly. Since, the nonlinearity is due to the product of input and output,this bilinear perceptron is considered as the simplified architecture of the class of fractionally spaced recursive polynomial perceptron, where the input signal vector have feedback sequences, input sequences and their product terms

We have examined and investigated the application of the polynomial perceptron and fractionally spaced recursive polynomial perceptron architectures to the problems in adaptive signal processing and show that these network architectures may be applied as a solution to noise compression in ECG signal

1.4 Back propagation

The back propagation is a learning law, the term is often used to refer to a hierarchical network architecture that uses back propagation updated algorithm for adapting the weights of each layer based on the error present at the network output The algorithm is a generalization of least mean square algorithm In its simplest structure, back propagation uses stochastic gradient search technique for the minimization of cost function It requires continuous differentiable non-linear function such as hard limiter function, sigmoidal function etc Some times when faster convergence rate is required, a momentum term is added to make the weight changes smooth

1.5 Organisation of Thesis

The Thesis is organised as follows

In present *chapter 1*, we present brief introduction of medical telemetry, artificial neural networks, review of various architectures and back propagation

In *chapter 2*, we present the architecture of complex neuron and the corresponding algorithm for the implementation of complex mapping from input to output. This complex structure of neuron is used in subsequent chapter for the complex domain mapping of input and output of polynomial perceptron architecture

In *chapter 3*, we present the mathematical concept of polynomial perceptron and fractionally spaced polynomial perceptron with their basic properities, features and limitations The subsequent section of this chapter cover the type of activation function and their need for implementation The last section of this chapter includes the algorithms which are used for the implementation of two dimensional complex domain representation

of polynomial perceptron and fractionally spaced polynomial perceptron

In *chapter 4*, we present the simulation results obtained by the implementation of back propagation algorithm, complex back propagation algorithm, polynomial perceptron and fractionally spaced bilinear perceptron We have applied the polynomial perceptron and fractionally spaced bilinear perceptron to ECG signal for noise suppers on using 16-level quadrature amplitude modulation systems

Chapter 2

Complex neuron structure

2.1 Introduction

In this chapter, we present the concept of complex neuron architecture and corresponding complex back propagation algorithm [17] It has been successfully applied in neural networks for classification and prediction It is generalized to include complex signals and coefficients for many nonlinear signal processing problems

We have used this nonlinear adaptive algorithm to our polynomial perceptron and fractionally spaced bilinear perceptron model, which require a complex mathematical representation of signal

2.2 Architecture

The basic element of multilayer [2] neural network is the neuron The multilayer perceptron state that the jth neuron in the mth layer has its primary local connections and is characterised by a set of real weights and real threshold. Then the output of the jth neuron may be written as :

$$y_j^{(m)} = f\left(\sum_{i=1}^{n} W_{ij}^{(m)} y_i^{(m-1)} + \theta_j^{(m)}\right) \tag{2.1}$$

where, $W_j^{(m)}$ and $\theta_j^{(m)}$ are the weights coefficient vector and the threshold for jth neuron of mth layer.

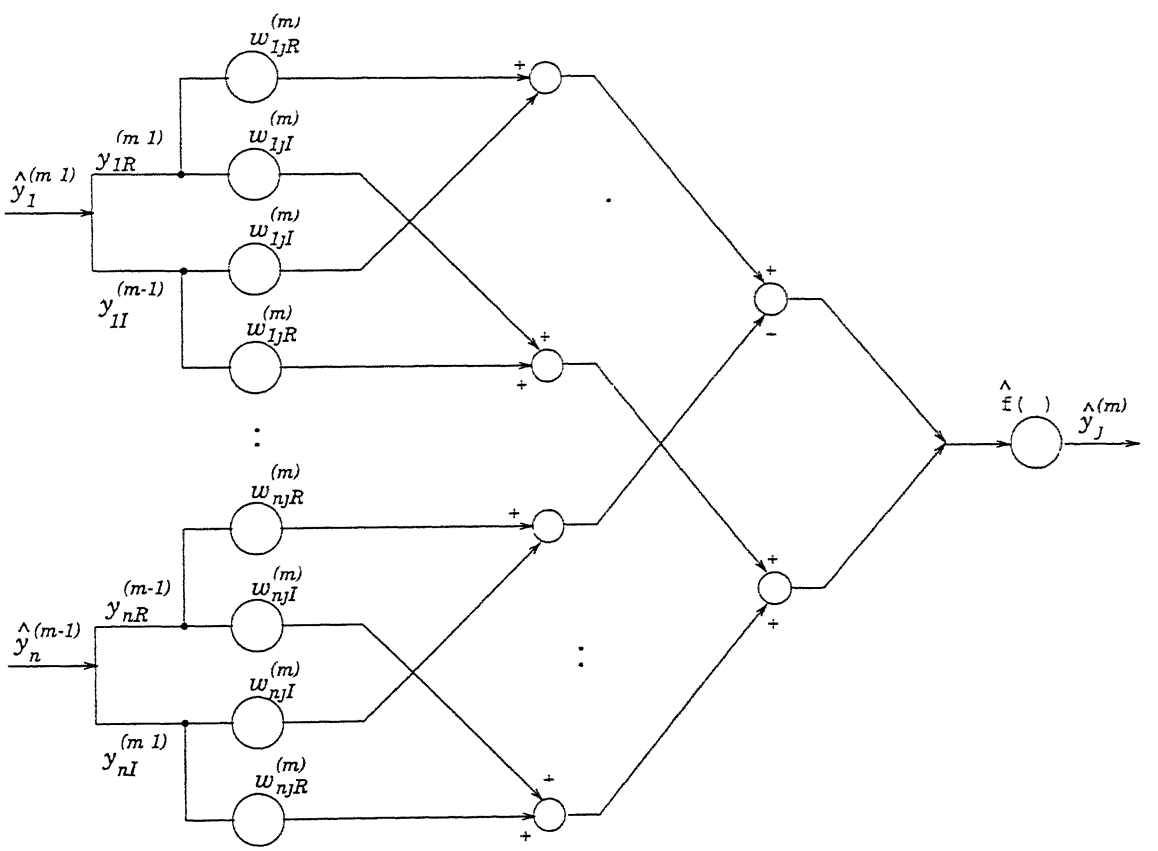


Fig 2.1: Architecture of complex neuron

$f(\ )$ is the nonlinear sigmoid type tangent hyperbolic activation function, which could be written as

$$f(x) = tanh(\gamma x/2) = \frac{1 - e^{-\gamma x}}{1 - e^{-\gamma x}} \tag{2.2}$$

This neuron architecture can be extended to complex domain as shown in fig 2.1 Similar to the the multilayer perceptron the complex output of the jth neuron in the mth layer can be written as follows

$$\begin{aligned}
\hat{y}_j^{(m)} &= y_{jR}^{(m)} + j y_{jI}^{(m)} \\
&= \hat{f}\left(\sum_{i=1}^{n}\left(W_{ijR}^{(m)} + W_{ijI}^{(m)}\right)\left(y_R^{(m-1)} + j y_I^{(m-1)}\right) + \left(\theta_{iR}^{(m)} + j\theta_I^{(m)}\right)\right) \\
&= \hat{f}\left(\left(\sum_{i=1}^{n}\left(W_{ijR}^{(m)} y_{iR}^{(m-1)} - W_{ijI}^{(m)} y_{iI}^{(m-1)}\right) + \theta_{jR}^{(m)}\right)\right. \\
&\quad \left. + j\left(\sum_{i=1}^{n}\left(W_{ijR}^{(m)} y_{iI}^{(m-1)} - W_{jI}^{(m)} y_{iR}^{(m-1)}\right) + \theta_{jI}^{(m)}\right)\right)
\end{aligned} \tag{2.3}$$

where, $\hat{f}(x + jy)$ is a complex mapping function, which can be defined as follows

$$\begin{aligned}
\hat{f}(x + jy) &= \frac{1 - e^{-\gamma x}}{1 + e^{-\gamma x}} + j\frac{1 - e^{-\gamma y}}{1 + e^{-\gamma y}} \\
&= f(x) + jf(y)
\end{aligned} \tag{2.4}$$

The γ is the slope parameter and the sign $\hat{}$ represents that the variable is complex

The structure of this complex neural architecture is similar to that of the multilayer perceptron It also consists of several hidden layers and is capable of performing complex mapping between input and output layer The difference is only that the neurons and the corresponding learning algorithms are in complex domain

2.3 Algorithm

This algorithm performs an approximation to the global minimization achieved by the method of the smoothed stochastic gradient Here, nonlinear sigmoid type function shown by equation (2.4), is used as the activation function The first order derivation of this complex function may be defined as follows

$$\begin{aligned}\hat{f}'(x+jy) &= \hat{f}'(x)+j\hat{f}'(y)\\ &= \frac{1}{2}\left(1-f^2(x)\right)+j\frac{1}{2}\left(1-f^2(y)\right)\end{aligned} \tag{2.5}$$

The error signal (ϵ_i), required for adaption, is defined as the difference between the desired response and the output of the perceptron

$$\epsilon_i(t) = d_i(t) - y_i(t) \qquad i = 1, 2 \quad \ldots N_m \tag{2.6}$$

where, d_i is the desired response at the ith node of the output layer. Hence, the sum of error squares produced by the network is as follows

$$E(t) = \sum_{i=1}^{N_m} \epsilon_i(t)\epsilon_i^*(t) \tag{2.7}$$

Here, we have used following steps for the implementation of this training algorithm with nonlinear sigmoid type complex activation function .

Step 1. Initially assign the small complex random values to all weight coefficients and node offsets

Step 2. Present input vector and desired response for all traing patterns

Step 3. By using the equation (2.4), calculate the outputs $y_1, y_2, \quad , y_{N_m}$ for all patterns. N_m are the number of nodes in the mth layer

Step 4. Adapt the weight coefficients as follows

$$\Delta_{ij}(t+1) = \alpha\Delta_{ij}(t) + \eta\delta_i x_j^* \tag{2.8}$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta_{ij}(t+1) \tag{2.9}$$

where α, and η are the momentum constant and training rate. x_j^* is the conjugate of the output of node j or input to node i and

$$\delta_i = \begin{cases} (d_i - y_i)f^{*\prime} & \text{for output layer} \\ f^{*\prime}\sum_k w_{ik}^* & \text{for input and hidden layers} \end{cases} \tag{2.10}$$

where, sign * represents the complex conjugate term

Step 5 Repeat by going to step 2

The generalization of the back propagation to deal with complex signals make it possible to use this powerful nonlinear signal processing algorithm

Chapter 3

Polynomial perceptron

3.1 Introduction

The polynomial systems [10] are those nonlinear systems whose output signals can be related to input signals through a truncated volterra series expansions The volterra series expansion can model a large class of nonlinear systems This is the basic idea behind the development of polynomial perceptron, which is able to approximate a large class of nonlinear signals within the specified accuracy

In this chapter we will discuss the architecture of polynomial perceptron and fractionally spaced recursive polynomial perceptron [11] in detail with their nonlinear mapping ability, features and limitations The polynomial function with sufficient size of degree can approximate any continuous function within a specified accuracy. It is shown here, that nonlinear mapping ability of polynomial perceptron is similar to multilayer perceptron with one hidden layer The performance of fractionally spaced recursive polynomial perceptron is better than polynomial perceptron with low complexity and fast convergence rate

The type of activation function, with their needs and the training algorithm used for the implementation polynomial perceptron as well as fractionally spaced recursive polynomial perceptron, are discussed in section 4 and section 5

3.2 Architecture of polynomial perceptron

The use of polynomial function to approximate a continuous function is an old but effective technique and is widely applied to the identification of nonlinear systems Any continuous function can be approximated to within an arbitrary accuracy by a polynomial function with a sufficient size If the polynomial function is directly used to approximate a nonlinear function, the degree of function has to be sufficiently high to achieve the adequate accuracy Therefore, complexity of polynomial perceptron is determined by the two parameters, namely the order (size of input vector) and the polynomial degree.

3.2.1 Properties

The architecture of polynomial perceptron with input order $m = 3$ and degree of polynomial $l = 2$ is shown in fig 3 1

Following polynomial decision function can be used as an approximate realisation of decision function

$$\begin{aligned} P_W^l(X(t)) &= w_0 + \sum_{i_1=1}^{m} w_{i_1} x(t - i_1 + 1) \\ &\quad + \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} w_{i_1 i_2} x(t - i_1 + 1) x(t - i_2 + 1) \\ &\quad + \cdot \quad + \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} \quad \sum_{i_l=i_{l-1}}^{m} w_{i_1 i_2 \quad i_l} x(t - i_1 + 1) \\ &\quad x(t - i_2 + 1) \quad x(t - i_l + 1) \end{aligned} \tag{3.1}$$

where,

$$X(t) = [x(t), x(t-1), \quad , x(t-m+1)]^T \tag{3.2}$$

is the m dimensinal input signal vector, and

$$W = [w_0, w_1, w_2, \cdot\cdot \ , w_m, w_{11}, w_{12}, \ \cdot\cdot, w_{11 \ 1}, \cdot\cdot \ , w_{mm \ m}]^T \tag{3.3}$$

is the n_k dimensional coefficient vector n_k may be obtained by the following equation ·

$$n_k = \sum_{i=0}^{l} n_i, n_0 = 1, n_i = n_{i-1}(m + i - 1)/i, i = 1, 2, \quad ., l \tag{3.4}$$

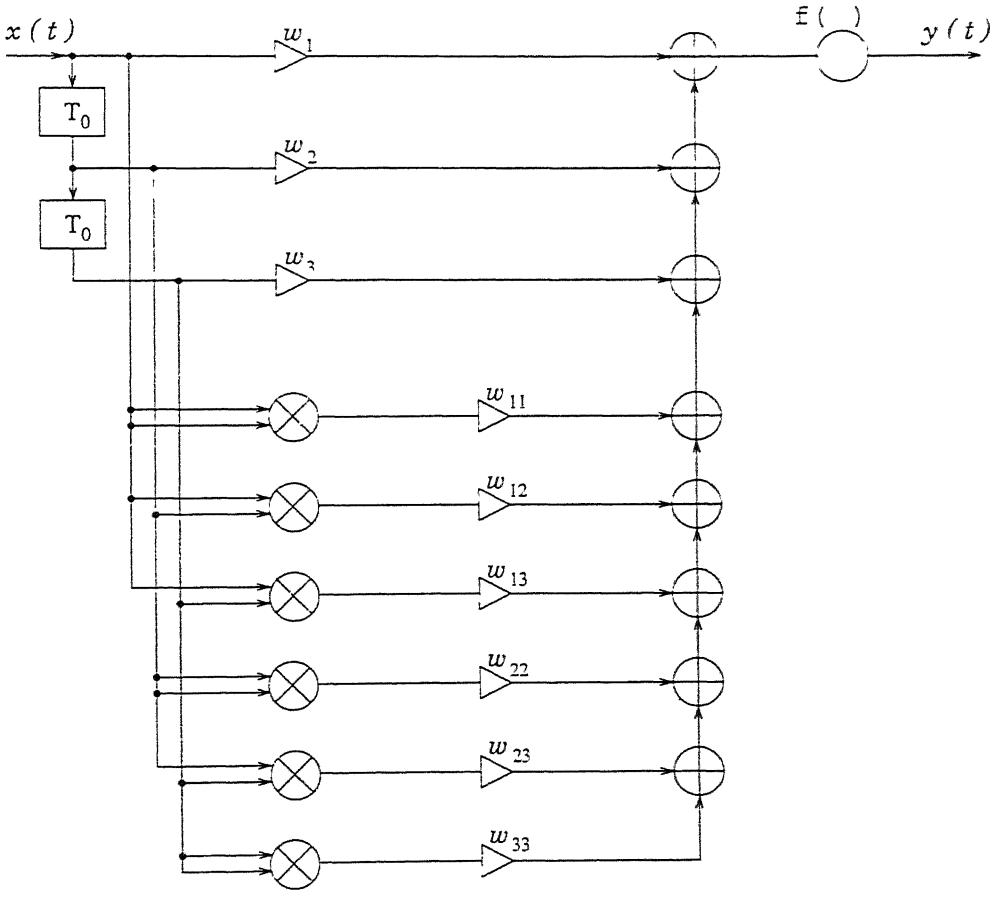


Fig 3 1 Architecture of polynomial perceptron

The mapping ability of polynomial perceptron can be shown by using the *Stone-Weierstrass Theorem* [12], which establishes the necessary and sufficient condition that a polynomial function can approximate any continuous real valued function

The equation (3.1) can be written as

$$P_W^l(X(t)) = \sum_{i=0}^{l} p_W^i(X(t)) \tag{3.5}$$

where,

$$\begin{aligned} p_W^0 &= w_0 \\ p_W^1(X(t)) &= \sum_{i_1=0}^{m} w_{i_1} x(t-i_1+1) \end{aligned}$$

$$\begin{aligned} p_W^l(X(t)) &= \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} \cdots \sum_{i_l=i_{l-1}}^{m} w_{i_1 i_2 \cdots i_l} x(t-i_1+1) \\ &\quad x(t-i_2+1) \cdots x(t-i_l+1) \end{aligned}$$

The equation (3.5) satisfies the following properties of *Stone-Weierstress Theorem*

- *Identity Function* It is obvious that the set of all polynomial functions $\{f(P_W^i(X(t))\}$, $i = 0, 1, \cdots, l$, contains constant functions $p_W^0(X(t)) = w_0$

- *Separability* · Take two distinct points $X(t)$ and $X'(t)$, we may assume that $x(t) \neq x'(t)$, then a degree-1 polynomial function $p_W^1(X(t))$ with a set of coefficients $w_0 = 0$, $w_1 \neq 0$, $w_i = 0$, $i = 2, 3, \cdots, m$ satisfies $p_W^1(X(t)) \neq p_W^1(X'(t))$.

- *Algebraic closure* The sum and product of two polynomial functions $P_{W_1}^{l_1}$ and $P_{W_2}^{l_2}$ are also the the polynomial functions The degree of polynomial for sum is the $max\{l_1, l_2\}$ and the degree of polynomial for product is $(l_1 + l_2)$ For example, if $l_2 > l_1$ then,

$$\begin{aligned} \left(P_{W_1}^{l_1} + P_{W_2}^{l_2}\right)(X(t)) &= \sum_{i=0}^{l_1} \left(p_{W_1}^i + p_{W_2}^i\right)(X(t)) \\ &\quad + \sum_{l_1+1}^{l_2} p_{W_2}^i(X(t)) \end{aligned} \tag{3.6}$$

$$\left(P_{W_1}^{l_1} \times P_{W_2}^{l_2}\right)(X(t)) = \sum_{i=0}^{l_1+l_2} p_W^i(X(t)) \tag{3.7}$$

where,

$$\begin{aligned}
\left(p^0_{W_1} + p^0_{W_2}\right)(X(t)) &= w_{1\,0} + w_{2,0} \\
p^0_W(X(t)) &= w_{1\,0} w_{2\,0} \\
\left(p^1_{W_1} + p^1_{W_2}\right)(X(t)) &= \sum_{i_1=1}^{m} (w_{1,i_1} + w_{2,i_1})\, x(t - i_1 + 1) \\
p^1_W(X(t)) &= \sum_{i_1=1}^{m} (w_{1\,0} w_{2,i_1} + w_{1\,i_1} w_{2\,0}) x(t - i_1 + 1)
\end{aligned}$$

$$\begin{aligned}
\left(p^{l_1}_{W_1} + p^{l_2}_{W_2}\right)(X(t)) &= \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} \cdots \sum_{i_{l_1}=i_{l_1-1}}^{m} \\
&\quad \left(w_{1,i_1 i_2 \cdots i_{l_1}} + w_{2,i_1 i_2 \cdots i_{l_1}}\right) \\
&\quad x(t - i_1 + 1) x(t - i_2 + 1) \cdots x(t - i_{l_1} + 1) \\
p^{l_1+l_2}_W(X(t)) &= \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} \cdots \sum_{i_{l_1+l_2}=i_{l_1+l_2-1}}^{m} \\
&\quad (w_{1,i_1 i_2 \cdots i_{l_2}} w_{2,i_{l_2+1} \cdots i_{l_1+l_2}}) \\
&\quad x(t - i_1 + 1) x(t - i_2 + 1) \cdots x(t - i_{l_1+l_2} + 1)
\end{aligned}$$

It is clear, from above properties that the set of all polynomial functions is dense in real banach space of continuous functions on X In other words, polynomial function is satisfying the *Stone-Weierstrass theorem* Therefore, polynomial function can approximate any continuous function within the specified accuracy

Since we can expand the sigmoidal function $f(x)$ by the *Taylor series expansion* as follows

$$f(x) = \frac{1}{1 + e^{-x}} = k_0 + \sum_{i=0}^{\infty} k_{2i+1} x^{2i+1} \tag{3 8}$$

where,

$$\begin{aligned}
k_0 &= \frac{1}{2} \\
k_i &= -\frac{1}{2} \sum_{j=1}^{i} \frac{(-1)^j}{j!} k_{i-j}, \qquad i \geq 1
\end{aligned} \tag{3 9}$$

Therefore, the output $y(t)$ can be written as follows

$$y(t) = f\left(P^l_W(X(t))\right)$$

$$= k_0 + \sum_{i=0}^{\infty} k_{2i+1} \left(P_W^l(X(t)) \right)^{2i+1}$$

$$= \sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} \cdots \sum_{i_m}^{\infty} k'_{i_1 i_2 \cdots i_m} x_1^{i_1} x_2^{i_2} \cdots x_m^{i_m} \qquad (3.10)$$

where, k' denotes the coefficients and the equation (3.10) is nothing but the well known volterra series. Now we can show that the mapping ability of polynomial perceptron is similar to the multilayer perceptron with one hidden layer.

Consider a three layer neural network with one hidden layer having two input signals x_1 and x_2. Where μ_{ij} is the weight from ith input signal to jth neuron of the input layer ν_{jk} is the weight from jth neuron of input layer to the kth neuron of the hidden layer and ϵ_k is the weight from kth neuron of hidden layer to the neuron of output layer.

The output of the jth neuron of the input layer, the output of kth neuron of the hidden layer and the overall output of the neural network may be written as follows

$$O_j^{(1)} = f\left(\sum_{i=1}^{2} \mu_{ij} x_i \right), \qquad j = 1, 2 \qquad (3.11)$$

$$O_k^{(2)} = f\left(\sum_{j=1}^{2} \nu_{jk} O_j^{(1)} \right), \qquad k = 1, 2 \qquad (3.12)$$

$$f(O) = f\left(\sum_{k=1}^{2} \epsilon_k O_k^{(2)} \right)$$

$$= f\left(\sum_{k=1}^{2} \epsilon_k f\left(\sum_{j=1}^{2} \nu_{jk} f\left(\sum_{i=1}^{2} x_i \mu_{ij} \right) \right) \right) \qquad (3.13)$$

Therefore, by using the equation (3.8) the above output equations may be rewritten as follows

$$O_1^{(1)} = k_0 + \sum_{i=0}^{\infty} k_{2i+1} (\mu_{11} x_1 + \mu_{21} x_2)^{2i+1} \qquad (3.14)$$

$$O_2^{(1)} = k_0 + \sum_{i=0}^{\infty} k_{2i+1} (\mu_{12} x_1 + \mu_{22} x_2)^{2i+1} \qquad (3.15)$$

$$O_1^{(2)} = k_0 + \sum_{i=0}^{\infty} k_{2i+1} (\nu_{11} O_1^{(1)} + \nu_{21} O_2^{(1)})^{2i+1} \qquad (3.16)$$

$$O_2^{(2)} = k_0 + \sum_{i=0}^{\infty} k_{2i+1} (\nu_{12} O_1^{(2)} + \nu_{22} O_2^{(2)})^{2i+1} \qquad (3.17)$$

$$O = \epsilon_1 O_1^{(2)} + \epsilon_2 O_2^{(2)} \qquad (3.18)$$

The equation (3 18) may be written as follows

$$O = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} K_{ij} x_1^i x_2^j \qquad (3\ 19)$$

where, K is the coefficient The equation (3 19) is also the volterra series

Hence, it is clear that nonlinear mapping ability of polynomial perceptron is identical to the mapping ability of three layer perceptron with one hidden layer

It is seen that the number of weight coefficients increases exponentially in the implemantation of polynomial perceptron Therefore, the polynomial perceptron is computationally more complex than simple linear structure Increasing computation complexity and dimensionality is a common price for employing a nonlinear architecture However the structure and operation of polynomial perceptron is simpler than multilayer perceptron

3.2.2 Features

The polynomial perceptron possesses following important features

- The polynomial perceptron provides a simple method of determining weights for cross product terms and power terms
- The algorithm adjusts the weight coefficients of the polynomial on the basis of one pattern at a time
- Since weight coefficients are adjusted after each pattern, the polynomial perceptron is able to use new information as it becomes available
- The learning procedure is not iterative, learning is complete after each pattern has been observed only once This feature makes storage of training pattern unnecessary
- With minor modification to the basic adapt algorithm, the polynomial perceptron can disregard old data and therefore follow nonstationary statistics
- The shape of the decision surfaces can be made as complex as necessary, or as simple as desired

- The computational and storage requirements of polynomial perceptron increase only linearly with number of weight coefficients used
- The number of weight coefficients used can approach or even exceed the number of training patterns with no danger of the polynomial overfitting the data

3.2.3 Limitations

- The major limitation of polynomial perceptron is that the number of weight coefficients increase exponentially with polynomial degree (l)
- Another limitation associated with the polynomial perceptron is that any noise in the input signal to the adaptive filter will appear in the multiplicative fashion and will affect the performance Because of the cross product terms and higher power terms, the noise in input data gets amplified This will affect the process of convergence and the network will not be able to learn the input patterns if noise level is very high

3.3 Architecture of fractionally spaced recursive polynomial perceptron

Feedback concept is very useful in modeling of many nonlinear systems The bilinear systems are one of them which utilize the feedback concept, where the output sequences are also used as a part of input signal vectors The simple architecture of fractionally spaced recursive polynomial perceptron is basically the bilinear architecture. This architecture overcomes the drawbacks of polynomial perceptron It requires small number of weight coefficients as compared to polynomial perceptron and is equally capable of approximating any continuous function within the specified accuracy

3.3.1 Properties

The input-output relation for this architecture may be shown by the following equation

$$\begin{aligned} y(t) &= f\left(P_W^l(Z(t))\right) \\ &= f\left(\sum_{i=1}^{l} p_W^i(Z(t))\right) \end{aligned} \tag{3.20}$$

where, P_W^l is a degree-l polynomial function with weight coefficient vector W and

$$\begin{aligned} Z(t) = [&y(t-T_0), y(t-2T_0), \quad y(t-(n-1)T_0), \\ &x(t), x(t-\tau_0), \quad x(t-(m-1)\tau_0)]^T \end{aligned} \tag{3.21}$$

is the $(m+n-1)$-dimensional input signal vector with $\tau_0 < T_0$ The architecture consist of asynchronous feedforward filters and a synchronous feedback filters Variables τ_0 and T_0 are tap spacing Variables m and n are the tap numbers of feedforward part and feedback part respectively, and p_W^i is the ith order polynomial function Since the architecture of this fractionally spaced recursive polynomial perceptron is similar to the architecture of polynomial perceptron, therefore, we can show that this architecture is also capable of approximating any continuous function within an arbitrary accuracy like polynomial perceptron as shown in previous section

Inspite of its simplicity, it is an important nonlinear architecture which can model the large class of nonlinear systems with few number of weight coefficients If the input signal vector space is compact, then, the set of all fractionally spaced recursive polynomial perceptrons $\{f(P_W^l(Z(t)))\}$ is dense in real banach space of continuous functions on Z

3.3.2 Fractionally spaced bilinear perceptron

The systems which are linear with respect to the input and output seperately, but nonlinear due to the products between input and output are known as bilinear systems [13]. The architecture of the bilinear system is shown in fig 3 2

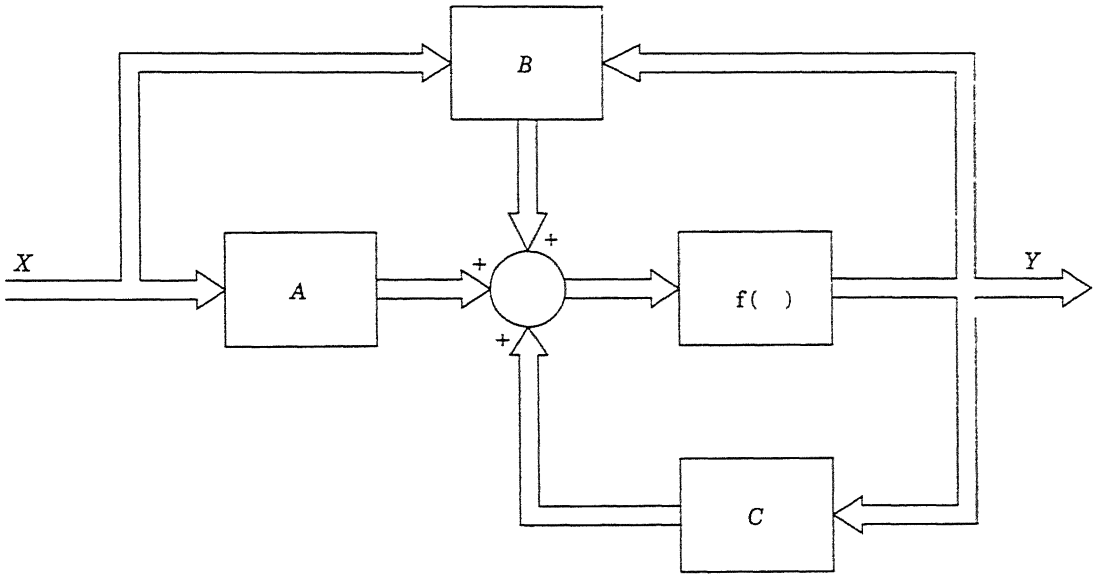


Fig. 3 2 Architecture of bilinear system.

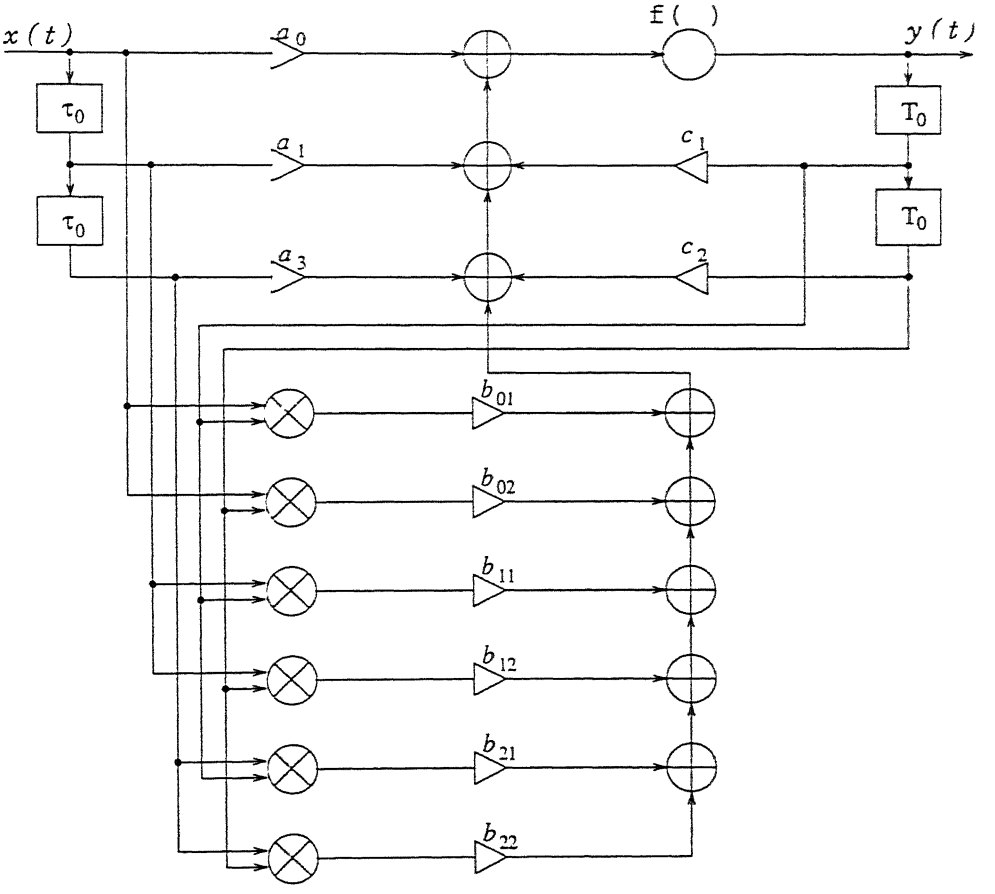


Fig 3 3· Architecture of fractionally spaced bilinear perceptron.

There are several theoretical and practical motivations for the importance attached to the class of bilinear systems This allows the application of a number of techniques and analytical procedures already set up for the linear systems The nonlinear structure of bilinear systems offers some important advantages over the linear case. Since the nonlinearity in the bilinear system is due to the product term the architecture of bilinear systems may be classified in the catagory of fractionally spaced recursive polynomial perceptron as fractionally spaced bilinear perceptron (FSBLP)

Fractionally spaced bilinear perceptron is the simplified architecture of fractionally spaced polynomial perceptron class The weight coefficient vector for fractionally spaced bilinear perceptron may be written as follows

$$W = [a_0, a_1, a_2, \quad , a_{m-1} \; c_1, c_2 \quad , c_{n-1} \; b_{01}, b_{02}, \quad , b_{(m-1)(r-1)}]^T \qquad (3.22)$$

Therefore, first and second degree polynomial functions may be written as follows

$$\begin{aligned} p^1_W(Z(t)) &= \sum_{i=0}^{m-1} a_i x(t-i) + \sum_{i=1}^{n-1} c_i y(t-i) \\ p^2_W(Z(t)) &= \sum_{i=0}^{m-1} \sum_{j=1}^{n-1} b_{ij} x(t-i) y(t-j) \end{aligned}$$

Then, the simplified output equation may be written as follows

$$\begin{aligned} y(t) = f\Bigg(& \sum_{i=0}^{m-1} a_i x(t-i) + \sum_{i=1}^{n-1} c_i y(t-i) \\ & + \sum_{i=0}^{m-1} \sum_{j=1}^{n-1} b_{ij} x(t-i) y(t-j) \Bigg) \end{aligned} \qquad (3.23)$$

where $f(\ )$ is the nonlinear sigmoid type activation function The *Taylor series expansion* of sigmoid function is given by equation (3.8)

The architecture of fractionally spaced bilinear perceptron with feedforward filter taps ($m = 3$) and feedback filter taps ($n = 3$) is shown in fig 3.3 The input to the feedforward filter is the received sequence and input to the feedback filter is the recovered sequence

3.3.3 Features

- This architecture possesses almost all the features of polynomial perceptron as shown above

- It requires smaller number of weight coefficients, which reduces the computational complexity and storage requirements

3.3.4 Limitations

- The obvious limitation of this representation is that it must continuously monitor for stability

- Another limitation associated with fractionally spaced recursive polynomial perceptron is that any noise in the input signal to the adaptive filter will appear in the multiplicative fashion and will affect the performance Because of the cross product terms, the noise in input data gets amplified This will affect the process of convergence and the network will not be able to learn the input patterns if noise level is very high

3.4 Activation Function

The activation function used for the implementation of polynomial perceptron and fractionally spaced recursive polynomial perceptron is the nonlinear function of the sigmoid type (tan hyperbolic function) as shown by the following equation

$$\begin{aligned} f(x) &= tanh(\gamma x/2) \\ &= \frac{1-e^{-\gamma x}}{1+e^{-\gamma x}} \\ &= \frac{2}{1+e^{-\gamma x}} - 1, \qquad \gamma > 0 \end{aligned} \tag{3.24}$$

This particular choice of the sigmoid function is because of the bipolar nature of the transmitted signal $x(t)$. Here γ is slope parameter The value of γ is normally selected as unity, but when the polynomial function is directly used to approximate the nonlinear function the value of γ is kept less than 1

Although, it is seen that the sigmoid function does complicate the mean square error (MSE) surface, yet the classifier will converge to the correct solution when initial parameters of training algorithm are inside a certain sphere.

The real criterion is the classification accuracy and the mean square error criterion is only for training to obtain an acceptable level of misclassification An alternative to do this is to increase the polynomial degree sufficiently, which increases the size of weight coefficient vector exponentially and increases the complexity Therefore by introducing the nonlinear sigmoid type activation function (tanh function), the polynomial degree can be restricted from 3 to 5, which would enable it to approximate the optimal solution

Since, we have considered the two dimensional classification problem therefore the neurons of the network, input sequences and output sequences are replaced by the their equivalent complex domain representation The sigmoid function is replaced by the complex mapping sigmoid function

By using the equation (2 4) the real and imaginary part may be defined as follows

$$\begin{aligned} f(u) &= \frac{1-e^{-\gamma u}}{1+e^{-\gamma u}} \\ &= \frac{2}{1+e^{-\gamma u}} - 1\,0 \qquad (3\,25) \\ f(v) &= \frac{1-e^{-\gamma v}}{1+e^{-\gamma v}} \\ &= \frac{2}{1+e^{-\gamma v}} - 1\,0 \qquad (3\,26) \end{aligned}$$

The equations (3 25) and (3 26) are used to obtain the output sequences for polynomial perceptron as well as for fractionally spaced polynomial perceptron

3.5 Training Algorithms

There are number of algorithms available to train the neural networks Here, we have used the smoothed stochastic gradient algorithm to obtain the best suitable weight coefficients for polynomial perceptron and fractionally spaced bilinear perceptron of networks.

3.5.1 Algorithm for polynomial perceptron

Step 1 Initially assign the small complex random values to all weight coefficients.

Step 2. Present input vector and desired training response

Step 3. Calculate the actual complex output sequence $\hat{y}(t)$ by using the equations (3.25) and (3.26) as given below

$$\hat{y}(t) = \hat{f}\left(P_W^l(\hat{X}(n))\right) \tag{3.27}$$

Step 4. Update the weight coefficients by using the smoothed stochastic gradient algorithm as follows

$$\Delta_i(t+1) = \alpha\Delta_i(t) + \eta\hat{e}(t)\left(1-\hat{y}^2(t)\right)\hat{M}_i(t) \tag{3.28}$$

$$\hat{w}_i(t+1) = \hat{w}_i(t) + \Delta_i(t+1) \tag{3.29}$$

$$i = 1, 2, \quad , n_k \tag{3.30}$$

where, the constant n_k is given by equation (3.4) The constants η and α are used as a adaptive learning rate and momentum constant, respectively $\hat{w}_i$ are the corresponding complex weight coefficients and Δ_i are complex difference element used to modify the weight coefficients $\hat{M}_i$ are the monomials of $\hat{x}(t), \hat{x}(t-1), \quad ., \hat{x}(t-m+1)$ from degree-0 upto degree-l

$$\hat{e}(t) = \hat{d}(t) - \hat{y}(t) \tag{3.31}$$

$\hat{e}(t)$ is the difference of desired training sample and the recovered output $\hat{d}(t)$ is the desired sample

Step 5. Repeat by going to step 2

3.5.2 Algorithm for fractionally spaced bilinear perceptron

Step 1. Initially assign the small complex random values to all weight coefficients

Step 2. Present input vector and desired training response

Step 3. Calculate the actual complex output sequence $\hat{y}(t)$ by using the equations (3.25) and (3.26) as given below

$$\hat{y}(t) = \hat{f}\left(\sum_{i=0}^{m-1}\hat{a}_i\hat{x}(t-i) + \sum_{j=1}^{n-1}\hat{c}_i\hat{y}(t-j) + \sum_{i=0}^{m-1}\sum_{j=1}^{n-1}\hat{b}_{ij}\hat{x}(t-i)\hat{y}(t-j)\right) \tag{3.32}$$

Step 4. Update the weight coefficients by using the smoothed stochastic gradient algorithm as follows

$$\Delta_{\hat{a}_i}(t+1) = \alpha\Delta_{\hat{a}_i}(t) + \eta_{\hat{a}}\hat{e}(t)\left(1-\hat{y}^2(t)\right)x(t-i) \quad (3.33)$$

$$\Delta_{\hat{c}_j}(t+1) = \alpha\Delta_{\hat{c}_j}(t) + \eta_{\hat{c}}\hat{e}(t)\left(1-\hat{y}^2(t)\right)y(t-j) \quad (3.34)$$

$$\Delta_{\hat{b}_{ij}}(t+1) = \alpha\Delta_{\hat{b}_{ij}}(t) + \eta_{\hat{b}}\hat{e}(t)\left(1-\hat{y}^2(t)\right)x(t-i)y(t-j) \quad (3.35)$$

$$\hat{a}_i(t+1) = \hat{a}_i(t) + \Delta_{a_i}(t+1) \quad (3.36)$$

$$\hat{c}_j(t+1) = \hat{c}_j(t) + \Delta_{c_j}(t+1) \quad (3.37)$$

$$\hat{b}_{ij}(t+1) = \hat{b}_{ij}(t) + \Delta_{b_{ij}}(t+1) \quad (3.38)$$

$i = 0, 1, 2, \quad , (m-1)$

$j = 1, 2, 3, \quad , (n-1)$

where, the terms $\eta_{(\,)}$ and α are used as adaptive learning rate and momentum constant, respectively $\hat{a}_i$, $\hat{c}_j$ and $\hat{b}_{ij}$ are the corresponding complex weight coefficients and Δ_i are the complex difference elements used to modify the weight coefficients

$$\hat{e}(t) = \hat{d}(t) - \hat{y}(t) \quad (3.39)$$

$\hat{e}(t)$ is the difference of desired training sample and the recovered output $\hat{d}(t)$ is the desired sample

Step 5. Repeat by going to step 2

The use of smoothed stochastic gradient algorithm improves the performance but it requires more computation in each recursion Therefore some times it can be replaced by stochastic gradient algorithm

The above algorithms are derived for two-dimensional input signal vector Therefore, the nodes are considered as complex neurons (as given in chapter 2) For one-dimensional input vector, same algorithm may be used by replacing the complex nodes by real nodes

Chapter 4

Simulation results

All of the simulations are performed on hp-850,convex C-220 akash and Dac-alpha machines

4.1 Back propagation algorithm

In this section we will present the results obtained from the implementation of the back propagation algorithm

The ability of back propagation algorithm is investigated using the classification problem of 16 sinwaves of different frequences

Since, the network is used as a classifier, therefore, all desired output are set to zero except for that corresponding class of inputs The desired output is set to one The following structure of multilayer perceptron is selected for this exercise

- Number of nodes in input layer = 128
- Number of nodes in output layers = 16.
- Number of hidden layer = 2
- Number of nodes in first hidden layer = 64
- Number of nodes in second hidden layer = 32

The training (or learning) rate (η) and momentum rate (α) are varied in step of 0 1 for 16 input samples of different sinewaves to obtain the number of iteration required for

restricting the global mean square error less than 0 03 The number of iterations required for differents combination of η and α are illustrated in fig 4 1 It is noted that, without momentum rate the convergence is very slow and network is not trained even in 300 iterations for any value of eta We have observed the following salient features

- The momentum rate plays an important role for the speed of convergence The convergence is faster if a small value of momentum rate is added
- The number of iterations decrease as the value of η increases
- The convergence is more uniform with momentum
- The number of iterations required for the convergence decreases with increasing α upto a particular limit, after that number of iterations increases for restricting the global error to the specified limit It is noted that if the α and the η both are greater than 0 6, the convergence is slow

In next step, the network is trained for these 16 sinewaves of different frequencies, with $\eta = 0\ 6$ and $\alpha = 0\ 3$ The fig 4 2 illustrates the reduction in mean square error for first 50 iterations during the training of network It is found that the mean square error is reducing smoothly and exponentially, with selected set of η and α The mean square error reached to less then 0 0004 within 50 iterations

Then, we have shifted these sinwaves and used them as the input for this trained network and found the corresponding outputs We have observed that this network is very powerful to identify these shifted sinewaves of differents frequencies It is noted that this particular network correctly identify these patterns upto the shift of 48 degree in the sine waves, beyond 48 degree shift it fails to identify these patterns

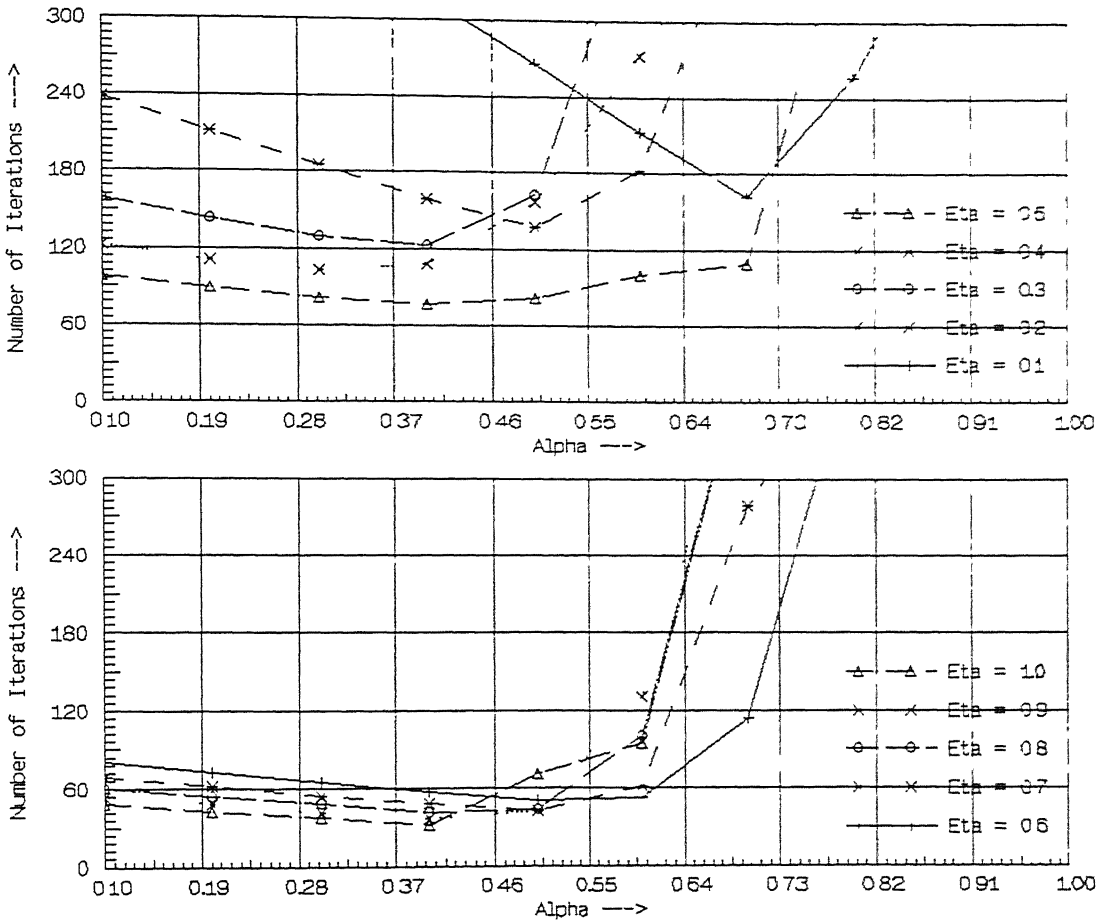


Fig. 4.1 Momentum rate versus number iterations

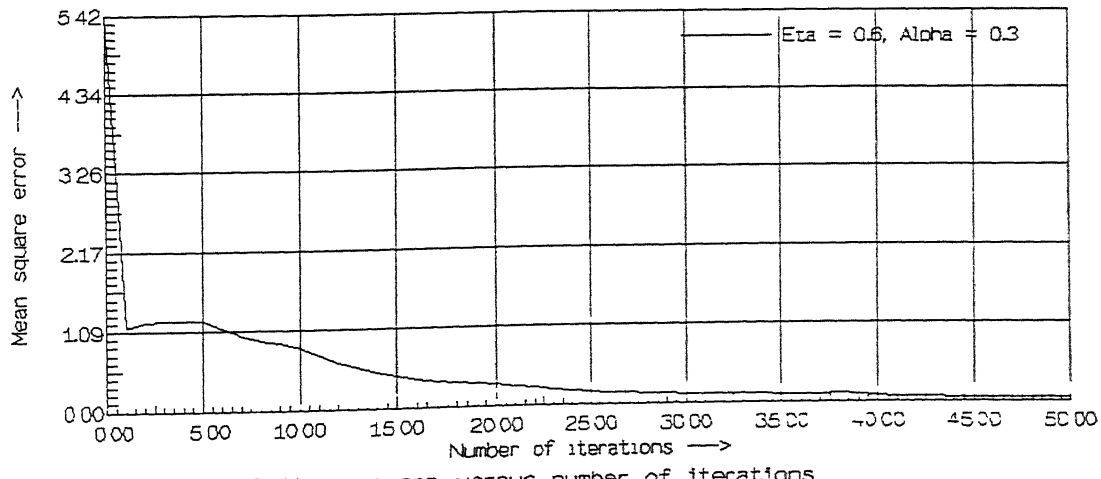


Fig 4.2 Mean square versus number of iterations

4 2 Complex back propagation algorithm

In this section we present the results obtained with the successful implementation of complex back propagation algorithm as given in chapter 2 In order to test the performance of this algorithm the following four input complex patterns are used for classification

- First input pattern [(1 1) (0 0) (0 0) (0 0)]
- Second input pattern [(0 0) (1 1) (0 0) (0 0)]
- Third input pattern [(0 0) (0 0) (1 1) (0 0)]
- Fourth input pattern [(0 0) (0 0) (0 0) (1 1)]

The network is trained for these patterns as input with two different configuration of multilayer perceptron

- In first case we have considered the {4 2 4} structure i e 4 input nodes 4 output nodes and one hidden layer with 2 nodes
- In second case we have considered the {4 8 4} structure i e 4 input nodes 4 output nodes and one hidden layer with 8 nodes

In both cases the learning rate (η) and momentum rate (α) are set to 0 6 and 0 1 respectively

The training results are obtained after 100 iterations 1000 iterations 2000 iterations and 4000 iterations for both structures {4 2 4} and {4 8 4} and illustrated in *table 1 1* and *table 1 2*

It is observed that the complex neuron architecture given in chapter 2 is indeed capable of recognizing the complex signals It is to be noted that the selection of number of hidden layers and number of nodes in each hidden layer as well as the learning parameters are purely experimental

Target	*100 Iterations*	*1000 Iterations*	*2000 Iterations*	*4000 Iterations*
(1 1)	(0 7589 0 7353)	(0 9595 0 9873)	(0 9728 0 9909)	(0 9813 0 9934)
(0 0)	(0 0047 0 0029)	(0 0188 0 0000)	(0 0134,0 0000)	(0 0082 0 0000)
(0 0)	(0 1173 0 0627)	(0 0491 0 0266)	(0 0335,0 0186)	(0 0233 0 0130)
(0 0)	(0 0757 0 8086)	(0 0149 0 0170)	(0 0118,0 0103)	(0 0086 0 0064)
(0 0)	(0 0017 0 0011)	(0 0000 0 0004)	(0 0000 0 0002)	(0 0000 0 0002)
(1 1)	(0 8687 0 9832)	(0 9704 0 9846)	(0 9779,0 9893)	(0 9846 0 9928)
(0,0)	(0 0917,0 0008)	(0 0420 0 0033)	(0 0292 0 0027)	(0 0202 0 0020)
(0,0)	(0 1774 0 1317)	(0 0285 0 0135)	(0 0176,0 0088)	(0 0121 0 0062)
(0 0)	(0 1092 0 2147)	(0 0320 0 0105)	(0 0219 0 0075)	(0 0151 0 0054)
(0 0)	(0 0025,0 0103)	(0 0098 0 0272)	(0 0092 0 0156)	(0 0068 0 0097)
(1,1)	(0 8904 0 9454)	(0 9457 0 9764)	(0 9625 0 9833)	(0 9738 0 9883)
(0,0)	(0 4091,0 0232)	(0 0206,0 0000)	(0 0125 0 0000)	(0 0078 0 0000)
(0,0)	(0 2606,0 0890)	(0 0290,0 0088)	(0 0193,0 0064)	(0 0133 0 0047)
(0,0)	(0 1575,0 0256)	(0 0364 0 0081)	(0 0226 0 0059)	(0 0154,0 0043)
(0,0)	(0 0012 0 0453)	(0 0000,0 0005)	(0 0000,0 0003)	(0 0000 0 0002)
(1 1)	(0 4887,0 8969)	(0 9599,0 9810)	(0 9737 0 9874)	(0 9820,0 9914)

Table 4 1 Results for {4 2 4} multilayer perceptron using complex back propagatation algorithm

Target	*100 Iterations*	*1000 Iterations*	*2000 Iterations*	*4000 Iterations*
(1 1)	(0 9294 0 9773)	(0 9816 0 9935)	(0 9874 0 9958)	(0 9914 0 9972)
(0,0)	(0 0263 0 0028)	(0 0078 0 0007)	(0 0048 0 0006)	(0 0031 0 0005)
(0,0)	(0 0552 0 0278)	(0 0125 0 0049)	(0 0083 0 0034)	(0 0056 0 0023)
(0 0)	(0 0405 0 0093)	(0 0130 0 0043)	(0 0092 0 0032)	(0 0065 0 0023)
(0 0)	(0 0155 0 0103)	(0 0095 0 0045)	(0 0061 0 0029)	(0 0044 0 0019)
(1 1)	(0 9238 0 9858)	(0 9802,0 9950)	(0 9871 0 9971)	(0 9914 0 9982)
(0 0)	(0 0598 0 0018)	(0 0136 0 0003)	(0 0089 0 0003)	(0 0060 0 0002)
(0,0)	(0 0653 0 0153)	(0 0158 0 0033)	(0 0106 0 0022)	(0 0073 0 0015)
(0 0)	(0 0453 0 0129)	(0 0131 0 0036)	(0 0092 0 0020)	(0 0063 0 0011)
(0 0)	(0 0045 0 0295)	(0 0122 0 0046)	(0 0080 0 0022)	(0 0053 0 0011)
(1 1)	(0 9460 0 9728)	(0 9828 0 9931)	(0 9887 0 9957)	(0 9924 0 9972)
(0 0)	(0 0417 0 0029)	(0 0054 0 0010)	(0 0033 0 0013)	(0 0022 0 0018)
(0,0)	(0 0436,0 0132)	(0 0094 0 0037)	(0 0062 0 0029)	(0 0043 0 0023)
(0,0)	(0 0708 0 0042)	(0 0129 0 0005)	(0 0083 0 0004)	(0 0055 0 0004)
(0 0)	(0 0069 0 0494)	(0 0091,0 0082)	(0 0060 0 0047)	(0 0041 0 0029)
(1,1)	(0 9244 0 9895)	(0 9828 0 9964)	(0 9884 0 9972)	(0 9920 0 9979)

Table 4 2 Results for {4 8 4} multilayer perceptron using complex back propagation algorithm

4 3 Polynomial perceptron and fractionally spaced bilinear perceptron

In this section we present the results obtained by polynomial perceptron and fractionally spaced bilinear perceptron (FSBLP) as given in chapter 3 The one dimensional and two dimensional structure of these algorithms are used for two different exercises

The one dimensional structure of these algorithms are used to train the networks for the ECG signal Here the direct sampled values of ECG signal are used as input sequences i e input vector is consider is one dimensional therefore the algorithms as given in chapter 3 are used by replacing the nodes from complex to real

The recorded ECG signal used in our studies is the low Q signal The samples of this signal are taken at intervals of 5 msec i e signal is sampled at the rate of 200 samples per seconds The total length of ECG signal is 4096 samples The amplitude of ECG signal is normalized between -1 to $+1$ because the used nonlinearity is hyperbolic tangent function which has the output range from -1 to $+1$ The reason for selecting this nonlinearity is the bipolar nature of ECG signal

The following architectures of polynomial perceptron and fractionally spaced bilinear perceptron networks are selected to train them with ECG signal

- Polynomial perceptron
  - Number of feedforward taps = 8
  - Degree of polynomial = 4
  - Learning rate (η) = 0 6
  - Momentum rate (α) = 0 4
  - Slope parameter (γ) = 1
- Fractionally spaced bilinear perceptron
  - Number of feedforward taps = 8
  - Number of feedback taps = 4

- Learning rate (η) = 0 6
- Learning rate (η_b) = 0 6
- Learning rate (η) = 0 6
- Momentum rate (α) = 0 4
- Slope parameter ($_f$) = 1

Both of the above networks are trained with a small segment of first 200 samples of ECG signals using specified learning rate and momentum rate

The next segment of 600 samples after first 200 samples is used as the input sequences for the trained networks The results are plotted in fig 4 3 These results show that the output signals in both networks are very close to the input signals Fig 4 3(d) illustrates the relationship between global mean square error and number of training times for PP and FSBLP It shows that mean square error can be restricted to $-40db$ by using the FSBLP with $\alpha = 0\ 4$

In next step, $15db$ of random noise is added to ECG signal and trained the FSBLP network with a segment of first 1200 samples of this noisy ECG signal The learning parameters are selected as $\eta = 0\ 2$ and $\alpha = 0\ 01$ The number of feedforward and feedback taps are selected as 20 and 10 respectively Then another segment of 400 samples is selected as the input sequence to this trained network The fig 4 4 illustrates the resultant output of this network The recovered signal shows that the network could not retrive the P wave of ECG signal correctly

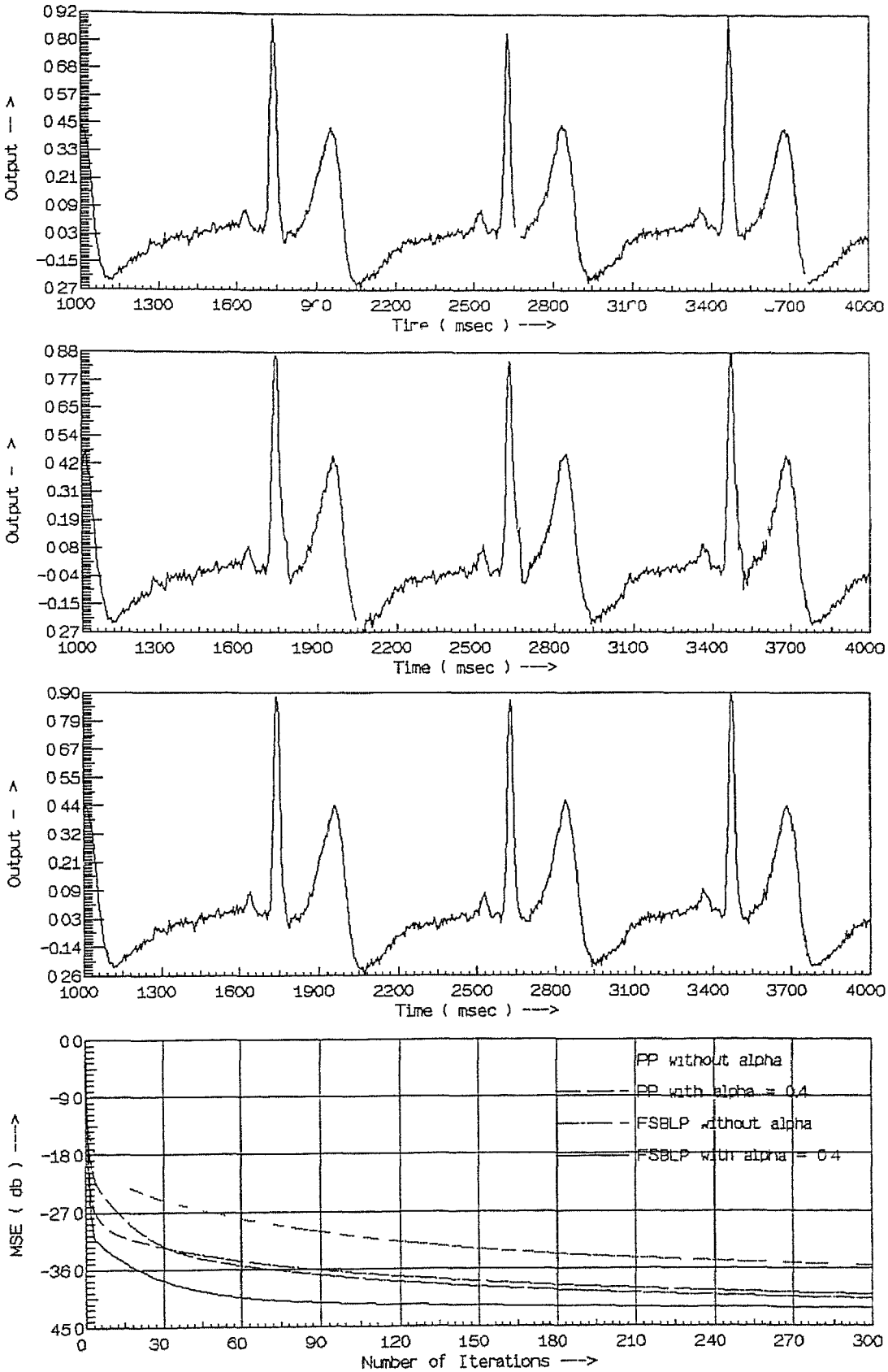


Fig. 4.3 (a) Original ECG signal (b) Output of PP (c) Output of FSBLP (d) MSE of {(8,4) PP} and {(8,4) FSBLP}

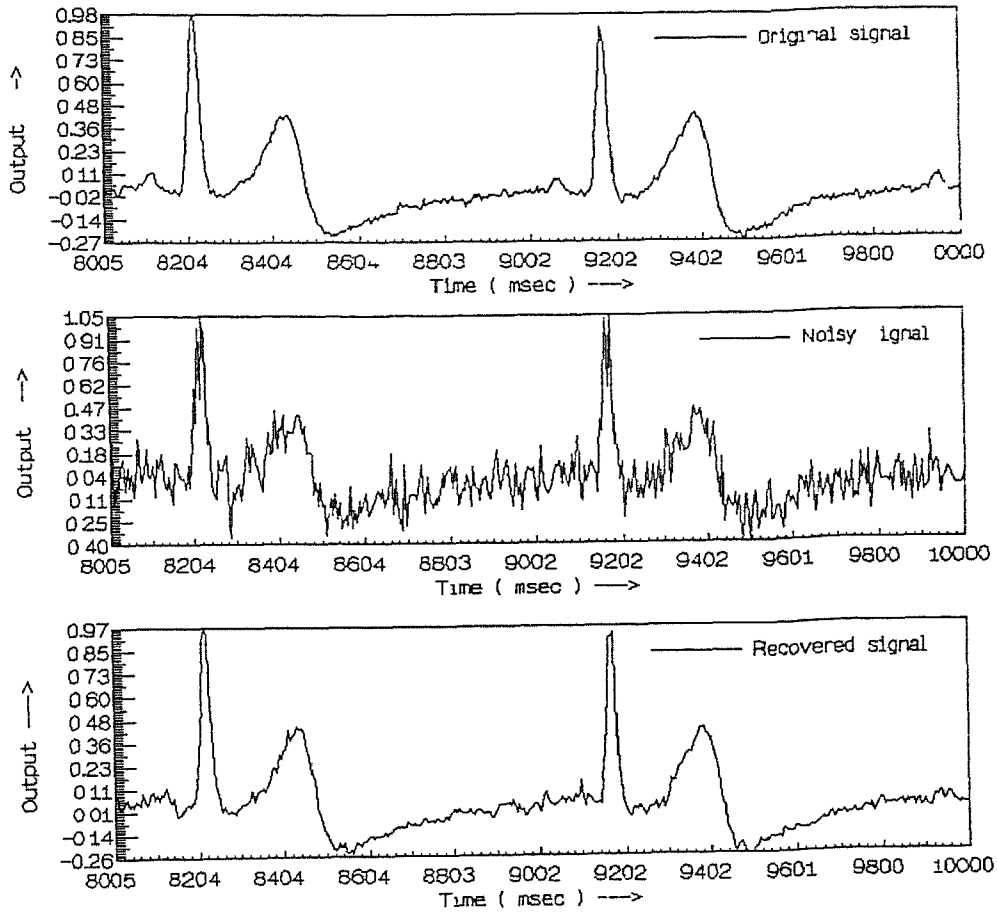


Fig 4.4 (a) Original ECG signal (b) Noisy ECG signal
(c) Recovered ECG signal

In next step we have investigated the behaviour of complex structure of polynomial perceptron and fractionally spaced bilinear perceptron The algorithms are implemented as given in chapter 3 and trained the networks to suppress the noise from 16 level quadrature amplitude modulation system

Quadrature amplitude modulation (QAM) Digital communication uses only a finite number of symbols for communication The minimum number of symbols are two in case of binary transmission When communication uses M symbols of different amplitude it is known as M ary QAM communication This multi amplitude signaling allows us to transmit each group of log_2M binary digits by one M ary pulse The block diagram of MQAM digital communication system is shown in fig 4 5 We have considered one practical case with M=16 i e the following 16 pulses are used for 16 different symbols

$$p(t) = a\, p(t) cos w\, t + b\, p(t) sin w_c t \qquad (4.1)$$

where, $p(t)$ is the transmitted signal $p(t)$ is a properly shaped baseband pulse, w is the carrier angular frequency and $[a(t) + jb(t)]$ is the equivalent complex baseband MQAM signal The choices of a and b for 16 pulses are shown in fig 4 6(a) Since M=16 therefore each pulse transmits the information of $log_2 16 = 4$ binary digits All the 16 different combination of (a, b) are illustrated in fig 4 6(a) Thus every possible 4 bit sequence is transmitted by a particular $(a \;\; b)$ Therefore one single pulse as represented by equation (4 1), will transmit the four bit information

The received signal r(t) may be written as follows

$$r(t) = Re\left\{[x_i(t) + jy(t)]e^{jw\ t}\right\} \qquad (4.2)$$

where, $[x(t) + jy(t)]$ is the equivalent complex baseband signal of r(t) This complex baseband signal may be written as follows

$$\begin{aligned} x(t) &= x_i(t) + jy_i(t) \\ &= [a(t) + jb(t)]R_1(t)e^{j\theta_1(t)} \\ &\quad + [a(t) + b_i(t)]R_2 e^{j(\theta_2(t) + w\ r)} \\ &\quad + [n_a(t) + jn_b(t)] \end{aligned} \qquad (4.3)$$

where $[R_1 e^{j\theta_1(t)} + R_2 e^{j(\theta_2(t)+w\ \tau)}]$ is two ray complex baseband model of frequency selective fading process on the transmitted signal and $[n\ (t) + jn_o\ (t)]$ is the complex baseband gaussian noise τ is the relative delay between main and reflected rays $R_i(t)$ s, i=1 2 are Rayleigh processes $R\ (t)$ s and $\theta\ (t)$ s are assumed to be statistically independent

It is assumed that the symbol recovery time is ideal x(t) is the sample of received complex baseband signal Fig 4 6(b) illustrates the noisy received samples These received samples are passed through the channel equalizer (neural network) before being mapped to the estimated signal vector

The following architecture of polynomial perceptron and fractionally spaced bilinear perceptron are used for the noise suppression from 16 QAM signals

- Polynomial perceptron
  - Number of feedforward taps = 20
  - Degree of polynomial = 4
  - Learning rate (η) = 0 0008
  - Momentum rate (α) = 0 001
  - Slope parameter (γ) = 1
- Fractionally spaced bilinear perceptron
  - Number of feedforward taps = 20
  - Number of feedback taps = 10
  - Learning rate (η_a) = 0 01
  - Learning rate (η_b) = 0 01
  - Learning rate (η_c) = 0 001
  - Momentum rate (α) = 0 001
  - Slope parameter (γ) = 1

In both cases, the algorithms are used to train the networks with 5000 input samples

The input samples are the noisy 16 level QAM signal where the random noise is added in such a manner that the samples can overlap the nearby patterns

In these algorithms the learning rate and the momentum rates are considerably small because the number of training samples and the number of weight coefficients in polynomial perceptron is too large The number of weight coefficients for above structure of polynomial perceptron is 10625 which is very large as compared to 209 for FSBLP therefore the learning rate and momentum rate for polynomial perceptron is chosen much less as compared to FSBLP

Initially, the networks are trained with these 5000 noisy samples Here the noise is added such that the samples may overlap upto 12 percent of nearby patterns and restricted the signal to noise ratio as low as $15db$

Another 15000 of noisy input samples with same noise level are used as input sequences for these trained network to find out the corresponding output sequences Fig 4 6(a) illustrates the ideal 16 QAM signal constellation where each pattern is represented by name P1 through P16 Fig 4 6(b) is illustrating the 15000 noisy samples These noisy received samples are used as the input for both networks The output results are illustrated in fig 4 6(c) and fig 4 6(d)

The output results of polynomial perceptron show that all of the signals belonging to patterns P1, P3, P11 P12 P13 P15 and P16 are mapped correctly but few of the signals from other patterns are mapped wrongly

The output results of fractionally spaced bilinear perceptron show that all signals belonging to patterns P1 P2 P6 P7 P10 P11 P12 P13 P15 and P16 are mapped correctly only very few of the signals from patterns P3 P4 P5 P8 P9 and P14 are mapped wrongly

These results show that 98 percent of correct samples are recovered from 12 percent of overlapped noisy signals by using the fractionally spaced bilinear perceptron with least probability of error

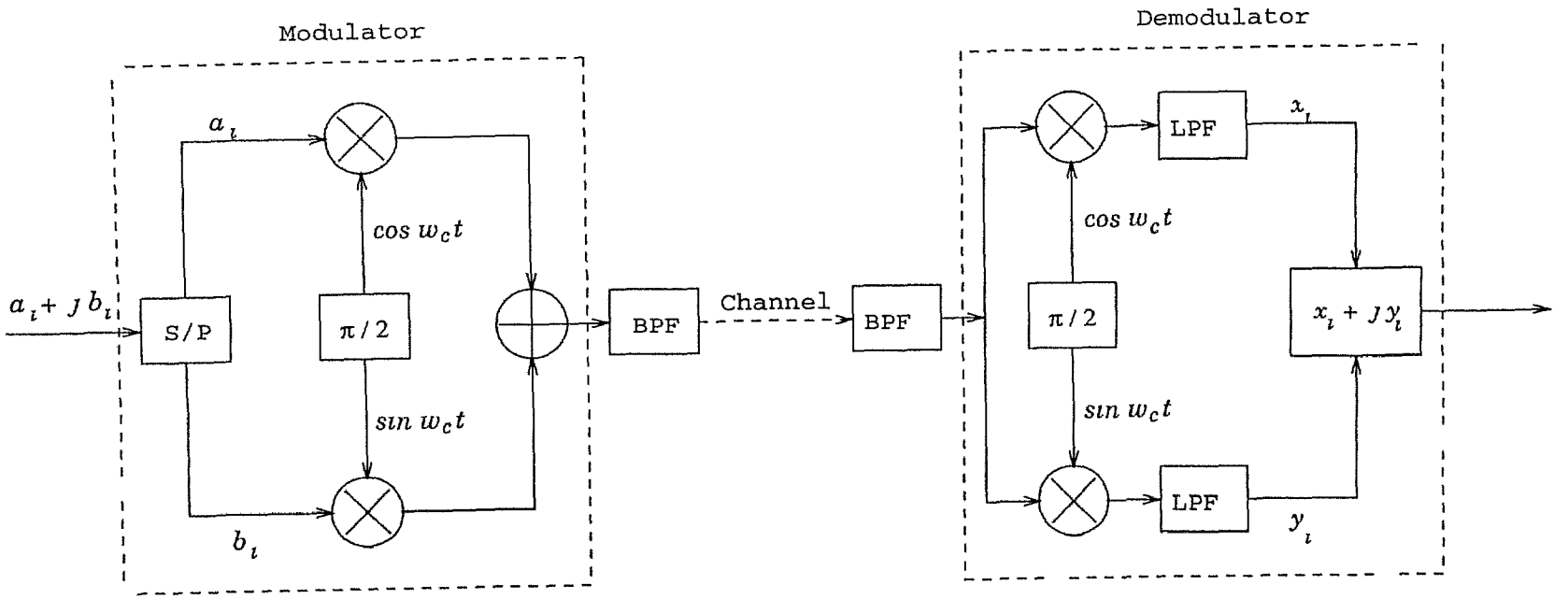


Fig 4 5 Block diagram of the MQAM digital communication systems

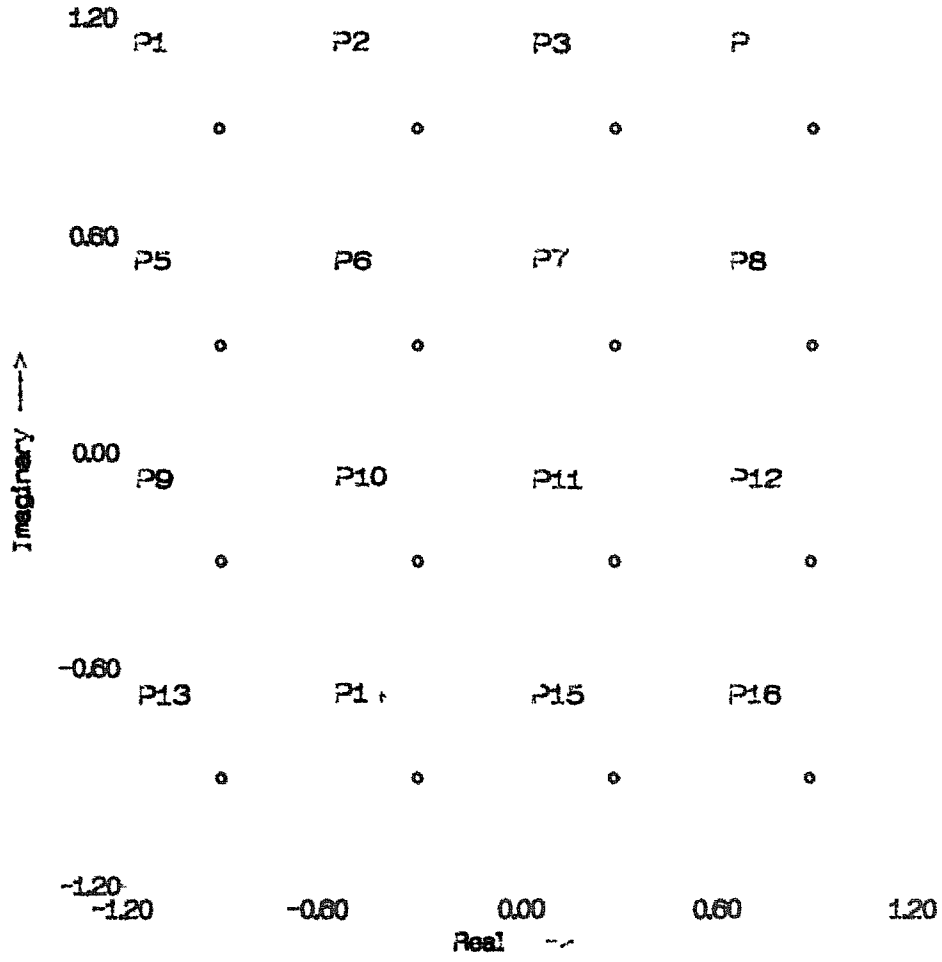


Fig.4.6(a) Ideal 16 Q- signal constellation

JS. 2.01

* saini * cc-iit/k tejas * 01/08/96 18:45.19 *

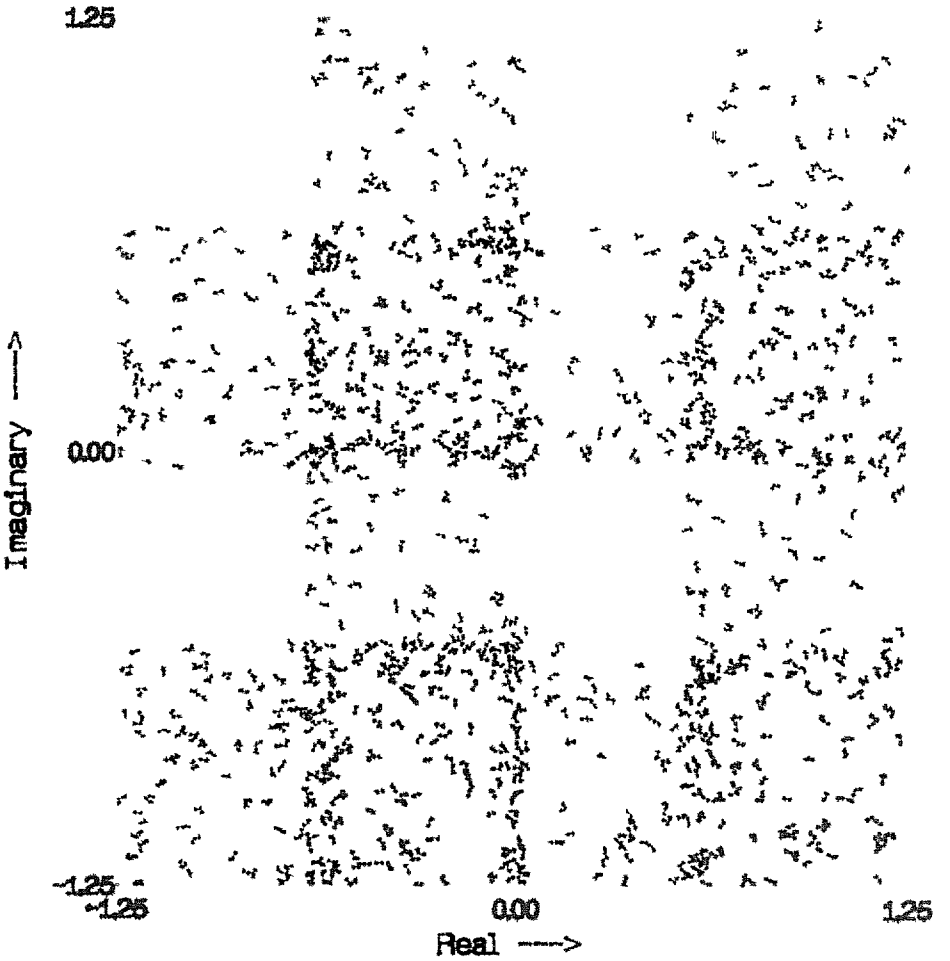


Fig.4.6(b) Noisy 16 QAM signal constellation

UsPlot

* saini * cc-iit/k tejas * 01/02/96 16:12:01 *

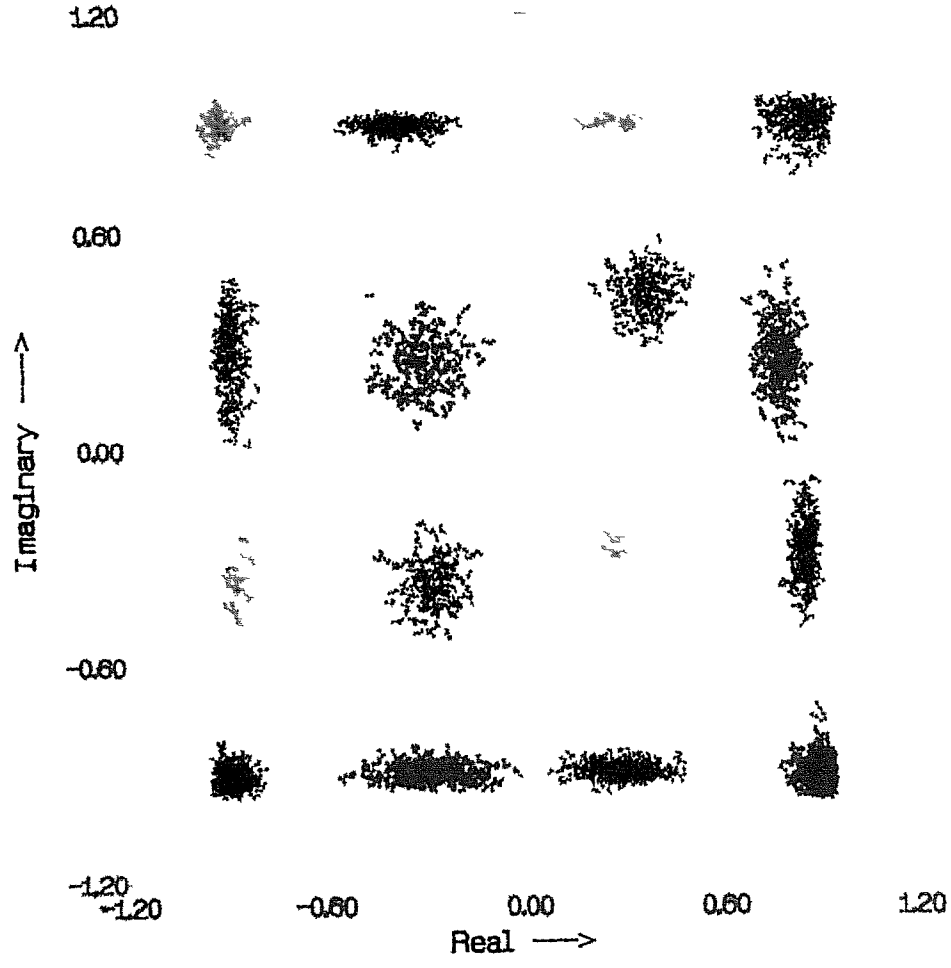


Fig 4.6(c) Recovered 16 QAM signal constellation by PP

UsPlot

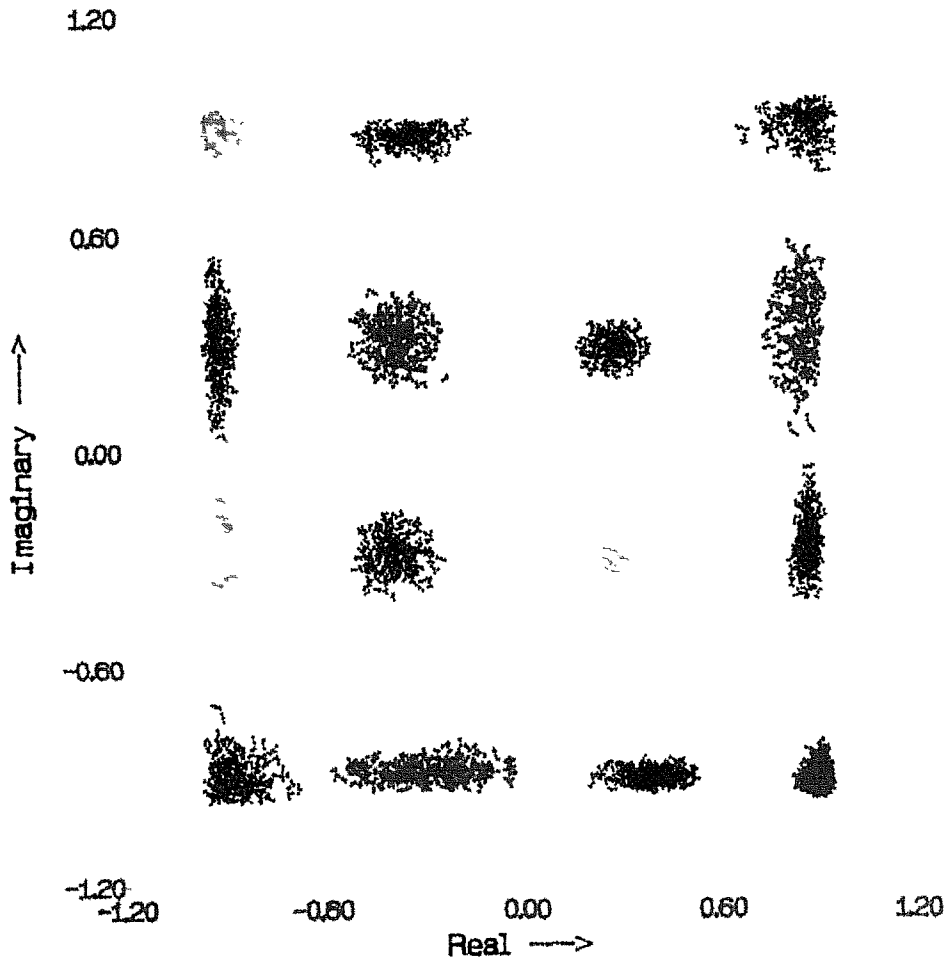


Fig 4.6(d) Recovered 16 QAM signal constellation by FSBLP

UsPlot

In next step FSBLP algorithm is used to suppress the noise and other channel interference from ECG signals Here the 16 level quadrature amplitude modulation system is used for adaptive equalization of random noise

The ECG signals are quantized into 12 bit convert each 12 bit sequence into 3 sequences of 4 bit each and send each 4 bits sequence sequentially in regular time interval

The $15db$ of random noise is again added to these 16 level QAM signal of ECG data The FSBLP network is then trained with a segment of first 1800 of these ECG data The following architecture of FSBLP is considered for this exercise

- Number of feedforward taps = 30
- Number of feedback taps = 15
- Learning rate (η) = 0 01
- Learning rate (η_b) = 0 01
- Learning rate (η) = 0 001
- Momentum rate (α) = 0 005
- Slope parameter (γ) = 0 2

In ECG signal after P Q delay the pacemaker pulse causes a sharp ventricular contraction (the QRS complex), therefore it becomes difficult to train the network with nonlinear sigmoidal function having the unity slope

It is observed that if the slope of sigmoidal function is reduced to 0 2 the network is trained with the desired accuracy

After completing the training another segment of next 4800 samples of these noisy signals is used as the input for this trained network Then using the simple thresholding the output sequences of FSBLP are mapped to their corresponding pattern ECG signal is reconstructed back from the threshold output sequences Fig 4 7 is illustrating the original ECG signal and the recovered ECG signal after suppression of the $15db$ of random channel noise

The recovered signal shows that the network could not retrive the first QRS complex, but after that rest of the part of signal is nicely recovered

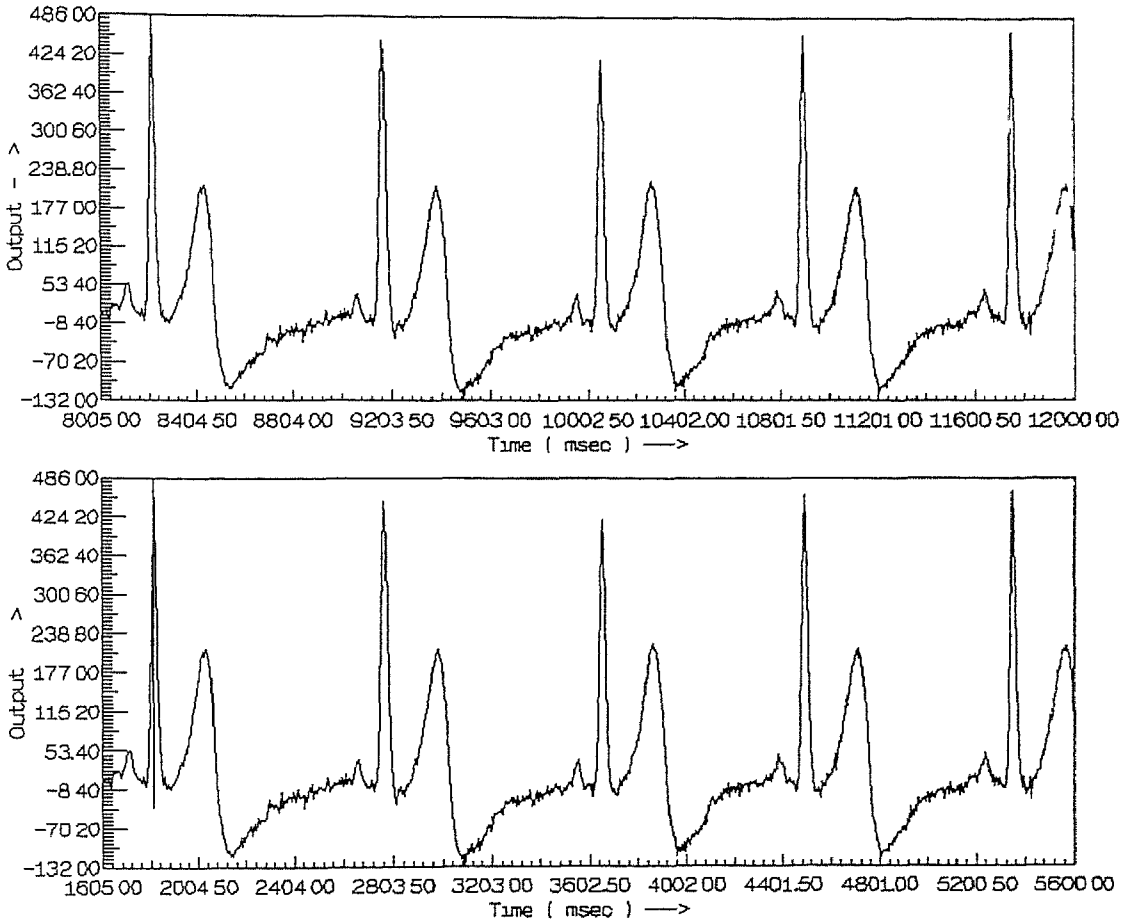


Fig 4 7 (a) Original ECG signal (b) Recovered ECG signal from 15db noise

Chapter 5

Conclusions

By viewing the equalization problem in long distance medical telemetry as a classification problems the influence of channel nonlinearities and additive noise to the optimal solution have been investigated It has been shown that the neural network approach offers equal effectiveness for adaptively equalizing the noise The backpropagation and complex back propagation algorithms are analized The complex neuron based polynomial perceptron and fractionally spaced polynomial perceptron algorithms are put forward for 16 level quadrature amplitude modulation system to recover the ECG signals from noisy environment Both of these algorithms are applied for 16 QAM system

Computer simulations have been carried out and results are compared Preliminary results obtained are satisfactory therefore following conclusions can be drawn

- The practical difficulties associated with back propagation and complex back propagation algorithms are the selection of number of hidden layers and number of nodes in each hidden layer The selection of learning parameters are experimental The rate of convergence can be increased by adding the momentum term in algorithms but simultaneously it increases the computational complexity

- The polynomial perceptron and fractionally spaced polynomial perceptron structured with nonlinear sigmoid type tangent hyperbolic activation function are capable of approximating the optimal solution The slope parameter of activation function play an important role in convergence Practically, it is found that the

slope parameter of 0 2 can train the network with the desired accuracy

- The complexity of the polynomial perceptron algorithm is determined by the two structure parameter, namely the order and degree of polynomial degree Practical selection of order and degree of polymomial has been discussed The polynomial perceptron with $4th$ degree is sufficient to approximate signals within the specified accuracy

- The structure of fractionally spaced bilinear is simple It possesses the fast con vergence rate and less computational complexity than polynomial perceptron The performance of fractionally spaced bilinear perceptron is better than that of poly nomial perceptron

- The fractionally spaced bilinear perceptron network recovered the 16 level QAM signals with less probability of error

- The ECG signal is retrived satisfactorily with fractionally spaced bilinear percep tron The network feels the difficulty only to retrive the first QRS complex, because in this technique, the weight updated is recurrent i e, weights are also updated during output generation During output generation, desired output response is substituted by its estimated value and the algorithm can continuously be employed to track a time varying environment

Bibliography

[1] D E Rumelhart G E Hinton and R J Williams Learning internal representations by error propagation in *Parellel distributed processing Exploration in the microstructure of cognition* vol 1 D E Rumelhart and J L McCelland Eds Cambridge M A Foundations M I T Press 1996

[2] R P Lippmann An Introduction to computing with neural network *IEEE ASSP Magazine* April 1987, pp 4 22

[3] H Leng and Simon Haykin, The complex backpropagation algorithm" *IEEE Trans on Signal processing* vol 39 no 9, September 1991 pp 2101 2104

[4] S Chen, G J Gibson and C F Cowan, "Adaptive channel equalisation using a polynomial perceptron structure" *IEEE Proc* , vol 137, Pt I no 5, October 1990 pp 257 264

[5] A C Tsoi and A D Back 'Locally recurrent globally feedback networks A critical review of architectures *IEEE Trans on Neural networks*, vol 5 no 2 March 1994 pp 229 239

[6] R J Williams and D Zipser 'A learning algorithm for continually running fully recurrent neural networks *Neural computation* vol 1, 1989 pp 270 280

[7] P Frasconi M Gori and G Soda Local feedback multilayered networks *Neural computation* vol 4 1992 pp 120 130

[8] D F Specht Generation of polynomial discrimiment functions for pattern recog nition *IEEE Trans on Electronic computers* vol EC 16 no 3 June 1967 pp 308 319

[9] D F Specht Probabilistic neural networks and the polynomial adaline as comple mentary techniques for classification *IEEE Trans on Neural Networks* vol 1 no 1 March 1990, pp 111 121

[10] V J Mathews Adaptive polynomial filters *IEEE SP Magazine* July 1991 pp 10 26

[11] Z Xiang G Bi and T Le Ngoc Polynomial perceptron and their applications to fading channel equalization and co channel interference suppression , *IEEE Trans on Signal processing* vol 42 no 9 September 1994 pp 2470 2480

[12] N E Cotter The stone weierstrass theorem and its application to neural networks *IEEE trans on Neural networks* vol 1 no 4 December 1990 pp 290 295

[13] C Bruni C Dipillo and G Koch 'Bilinear systems An appealing class of nearly linear systems in theory and applications , *IEEE trans on Automatica control* vol AC 19 no 4 August 1974, pp 334 348

[14] W J Thompkins Biomedical digital signal processing *Prentice Hall New Jersey* 1993

[15] J DuBovy 'Introduction to biomedical electronics' *McCraw Hill book company* 1978

[16] B P Lathi, 'Modern digital and analog communication systems" Second edition *Prism indian edition* 1993

[17] S Haykin "Neural network *Macmillan publishing company NJ* 1994

[18] J M Zurada, Introduction to artificial neural systems' *Jaico publishing house India*, 1994

EE-1996-M-SAI-POL